Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN SISTEMAS DE INFORMACIÓN

Visor cartográfico para la visualización masiva de trayectorias

Estudante: Eva Ocampo Quintáns
Dirección: Alejandro Cortiñas Álvarez
Dirección: Adrián Gómez Brandón
Dirección: Tirso Varela Rodeiro

A Coruña, setembro de 2020.

A mi familia.

Agradecimientos

En primer lugar, agradecer a mis tutores, Alejandro, Adrián y Tirso por darme la oportunidad de llevar a cabo este proyecto, por la paciencia y la dedicación que han tenido.

En segundo lugar, agradecérselo a mis padres, mis abuelos y mi hermana, por haberme animado a seguir adelante en todo momento y por haberme soportado durante estos duros meses de trabajo, que sé que en muchas ocasiones no se lo he puesto fácil. Especial mención a mi hermana, por la paciencia y apoyo.

Finalmente, a mis amigos por la confianza depositada en mí.

Resumen

El objetivo de este trabajo de fin de grado es desarrollar una aplicación web para la gestión de flota utilizando la estructura compacta Grammar based Compressed representation of Trajectories (GraCT), que permite obtener información histórica de los movimientos de los barcos, como obtener su trayectoria en un intervalo de tiempo, consultar los barcos más próximos a un punto, consultar los barcos que hay en una región y consultar la trayectoria de los barcos de una región en un intervalo.

Para alcanzar este objetivo fue necesario en primer lugar hacer un análisis preliminar para determinar el alcance y los requisitos funcionales del proyecto. A continuación, se llevó a cabo el diseño de prototipos de pantalla que daban soporte a las funcionalidades y la creación del modelo de datos, y finalmente se realizó la implementación de los requisitos e integración con la estructura GraCT y las pruebas necesarias.

En el desarrollo se empleó la tecnología MongoDB para el almacenamiento de información, así como las tecnologías Express.js y Node.js para la implementación del servicio REST. Se utilizó la tecnología Vue.js para la visualización junto con Leaflet para el desarrollo del visor de mapas, y las tecnologías N-API y C++ para la integración de la estructura compacta y el back-end.

El trabajo de fin de grado se gestionó siguiendo una metodología iterativa e incremental para el desarrollo de software.

Abstract

The aim of this project is to develop an application to manage fleet using the compact data structure Grammar based Compressed representation of Trajectories (GraCT), which allows to get historical information about position of the ships, like getting the trajectory between two time instants, getting nearest ships at a position, querying ships that lie in a region at time instant and querying the ships' trajectory in a region between two time instants.

In order to achieve this goal, it was necessary, first of all to do a preliminary analysis to resolve the scope of the project and its functional requirements. Next, we design different screen prototypes and the data model to support those functionalities. Finally, we implement the requirements and the integration of the compact data structure GraCT, with the required tests.

In the development, a technology MongoDB was used for the storage of information, as well as the technologies Express.js and Node.js to implement a REST service. The technology

Vue.js was used for visualization with Leaflet to develop the map viewer. Finally, N-API and C++ technologies was used to create the integration between GraCT and our back-end.

This work was managed following an iterative and incremental methodology for software development.

Palabras clave:

- GraCT
- Barco
- Trayectoria
- Instantes de tiempo
- Usabilidad
- Vue
- Leaflet
- Mongo
- Express
- C++
- N-API

Keywords:

- GraCT
- Ship
- Trajectory
- Time instants
- Usability
- Vue
- Leaflet
- Mongo
- Express
- C++
- N-API

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1 Motivación

Debido al creciente uso de información geográfica, se popularizó en la última década la inclusión de los GPS en casi todos los dispositivos de uso cotidiano, como pueden ser los barcos, aviones, coches; o incluso elementos de uso personal (relojes o móviles inteligentes). Como estos dispositivos están en constante movimiento es necesario tener un histórico de datos que permita conocer la trayectoria realizada o un punto en donde estuvo el objeto en algún instante de tiempo. Una trayectoria es un recorrido seguido por un objeto cuando se desplaza desde un punto de inicio a un punto de fin.

Con tal cantidad de datos geográficos fue necesario empezar a trabajar en distintas técnicas de compresión que permitan guardar la información usando el mínimo espacio necesario para acceder a ella de la forma más eficiente posible. Por este motivo, se llevó a cabo el desarrollo de la estructura compacta Grammar based Compressed representation of Trajectories (GraCT) [1]. Esta es una estructura que permite almacenar y consultar de forma eficiente las trayectorias libres de objetos en movimiento, es decir, trayectorias que no están sujetas a una red de rutas, como sucede en el caso de los barcos, aviones, etc. La estructura GraCT de datos ocupa menos que los datos comprimidos con un potente compresor tradicional, y además permite el acceso directo a las trayectorias de los objetos o a su posición en instantes de tiempo específico, así como rangos espaciales, consultas por proximidad en instantes o intervalos de tiempo concretos. Esto se puede enfocar al mundo de los barcos porque sus trayectorias no están restringidas a una red de rutas. A nivel de usuario es útil ya que GraCT ofrece consultas para saber la ruta que hizo un barco o cuáles son aquellos que están más próximos a un punto de interés, como un puerto o dentro de una zona de pesca de bajura.

No obstante, la estructura compacta GraCT se encuentra bastante alejada del usuario final, porque no proporciona una interfaz web con las funcionalidades necesarias para poder hacer estas consultas. Por lo que, el objetivo principal de este proyecto es desarrollar una aplicación

de gestión de flota que permita almacenar grandes cantidades de datos y facilite al usuario la realización de consultas sobre los barcos, utilizando por debajo la estructura GraCT.

La ventaja de esta aplicación es que usa una estructura compacta que aporta grandes beneficios a la hora de almacenar información de la aplicación permitiendo trabajar con gran cantidad de datos de forma eficiente, así como a la hora de realizar las consultas.

1.2 Objetivos

El objetivo principal de este proyecto es, por una parte, el desarrollo de una aplicación de gestión de flota que permita rastrear las trayectorias realizadas por los barcos, así como realizar consultas espacio-temporales y su posterior visualización en el visor de mapas. Por otra parte, que esta aplicación soporte de manera eficiente una gran cantidad de datos, para lo que se usa la estructura compacta en lugar de una base de datos tradicional, ya que esta no soportaría almacenar tantos datos ni consultarlos de forma eficiente y rápida como lo permite GraCT. Además esto posibilita acercar un prototipo de investigación como es GraCT al "mundo real".

A su vez este objetivo principal se divide en los objetivos secundarios que se exponen a continuación:

- Integración del visor de mapas con la estructura compacta GraCT. Uno de los objetivos secundarios más importantes es la comunicación del visor de mapas con GraCT, que permite que se realicen consultas de las trayectorias libres de los barcos en movimiento y de puntos de interés en instantes concretos de tiempo, así como también el almacenamiento de grandes cantidades de información geográfica de los barcos.

- Permitir consultas útiles para el dominio. Que la aplicación permita a los usuarios hacer consultas como obtener la trayectoria de un barco entre dos instantes de tiempo, obtener los barcos de una región en un instante de tiempo, obtener la trayectoria de los barcos de una región en un intervalo de tiempo y obtener los barcos más próximos a un punto.

- Visualización de información geográfica. Visualizar los datos geográficos que son resultado de las consultas realizadas por el usuario en el visor de mapas y que a este le resulte intuitivo. Los datos serán obtenidos de la estructura compacta GraCT, donde se encuentra almacenada la información de los recorridos de los barcos.

- Posibilidad de gestionar los usuarios con permisos especiales como el de administrador, para que se encargue de la gestión de los barcos y grupos de la aplicación, así como dar permiso de acceso, a los usuarios registrados, a cierta información de los barcos mediante la creación de grupos.

- Creación de reports. Los usuarios podrán hacer comentarios sobre los diferentes barcos que pertenezcan a sus grupos y estos se visualizarán solo para aquellos usuarios que también estén en los mismos grupos.
- Creación de una aplicación centrada en la usabilidad. Se pretende que la aplicación proporcione al usuario final un uso intuitivo de esta, así como obtener los resultados de manera rápida.

Capítulo 2

Fundamentos tecnológicos

2.1 Estado del arte

Se han estado buscado en el mercado herramientas que satisfagan los objetivos descritos en el Apartado 1.2 y se han encontrado las alternativas que a continuación se describen.

- **Marine Traffic** es una aplicación para la visualización de los movimientos de los barcos en tiempo real. En la Figura 2.1 se puede observar la página principal donde se muestra la última posición de los barcos recibida. Esta aplicación tiene diversas funcionalidades, entre ellas el filtrado de barcos en función de las diferentes características de estos, así como el viaje que llevaron a cabo, pero no la trayectoria, indicando puerto de origen y destino. También se pueden mostrar mapas del clima y mapas de densidad, además de poder consultar la información detallada de cada barco. Para poder tener acceso a algunas de estas funcionalidades es necesario realizar el pago de una mensualidad.

- **My Ship Tracking** esta aplicación también visualiza los movimientos en tiempo real de los barcos. Proporciona algunas de las mismas funcionalidades que proporciona la aplicación Marine Traffic, pero a diferencia de esta, no es necesario registrarse ni el pago de la mensualidad para realizar algunas consultas como el filtrado de los barcos indicando puerto de origen y destino, como se muestra en la Figura 2.2.

Ambas aplicaciones utilizan el sistema AIS, sistema automático para la identificación de embarcaciones. Es un sistema que provee las posiciones, rumbo y velocidad de las embarcaciones, mediante frecuencias VHF que son de alcance limitado. Permite ver a las embarcaciones como estaciones en alta mar si estas están equipadas tanto con el receptor como con el emisor AIS. Este sistema solo visualizará aquellas embarcaciones que estén equipadas con él.

El inconveniente de estas aplicaciones es que no comprimen, por lo tanto, no soportan una estructura compacta que pueda trabajar con una enorme cantidad de datos de manera

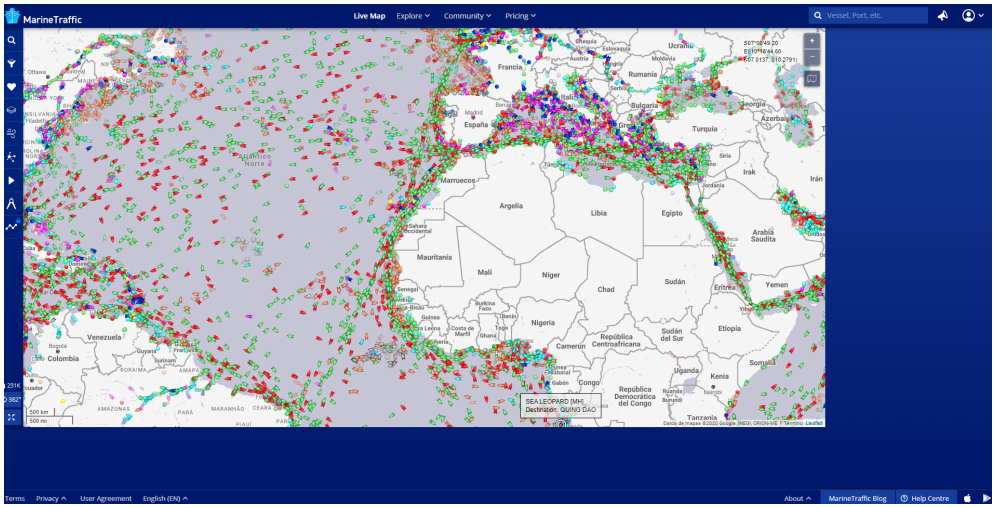


Figura 2.1: Inicio de Marine Traffic.

ágil. El contexto de estas aplicaciones es diferente al nuestro, ya que estas son de seguimiento de los movimientos de los barcos en tiempo real. Esto sería una desventaja frente a nuestro proyecto, ya que no almacenan datos históricos, permitiendo solamente hacer consultas de datos actuales. Además, como utilizan el sistema AIS, solo se muestran en el mapa aquellas embarcaciones que lo posean.

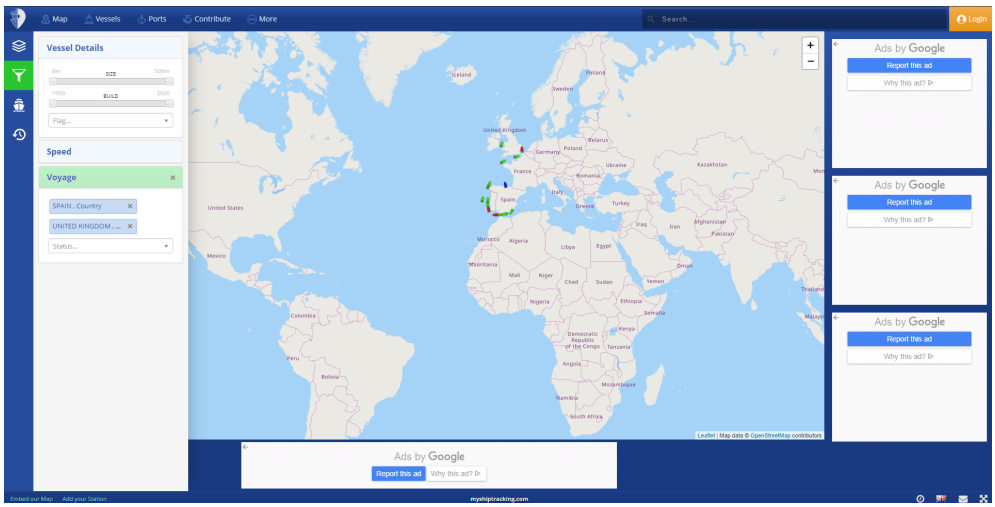


Figura 2.2: Filtrado de barcos por puerto origen y destino de viaje.

2.2 Tecnologías utilizadas

- Leaflet [2]. Es una librería de código abierto de JavaScript que permite crear mapas interactivos para la web de manera sencilla, proporcionando además gran variedad de plugins para extender su funcionalidad.

- Vue.js [3]. Es un framework progresivo de código abierto de JavaScript que permite crear interfaces de usuario. Este framework trabaja con componentes, lo que nos permite crear código reutilizable. Dentro de estos componentes se encuentran las etiquetas de HTML, el código JavaScript y el estilo CSS.

- JavaScript [4]. Es un lenguaje de programación que permite el desarrollo de páginas web y aplicaciones de servidor. Este lenguaje de programación es dinámico y soporta la construcción de objetos basados en prototipos. Se ha utilizado esta tecnología para la implementación de las funciones que comunican la vista con el modelo del cliente web.

- Axios [5]. Es una librería de JavaScript que está basada en promesas y que permite realizar peticiones HTTP desde un cliente web de manera sencilla.

- Node.js [6]. Es un entorno que trabaja en tiempo de ejecución, es de código abierto y permite a los desarrolladores crear herramientas en la parte del servidor y aplicaciones en JavaScript.

- Express.js [7]. Es un framework rápido de Node que permite crear APIs y aplicaciones web de manera sencilla. Proporciona mecanismos para realizar peticiones HTTP en diferentes URL, también para el procesamiento de peticiones *middlewares* adicionales como para el inicio de sesión de un usuario, cookies, etc. Un *middleware* es software que se ejecuta entre la petición que hace el usuario hasta que llega al servidor. También proporciona mecanismos para establecer ajustes de la aplicación como el puerto a utilizar y la realización de la conexión a la base de datos.

- MongoDB [8]. Es un sistema de base de datos NoSQL basado en documentos (JSON/BSON). Permite crear colecciones y cada una de ellas está formada por documentos que son objetos *json*. Se pueden crear esquemas para definir los documentos, proporcionando flexibilidad en ellos ya que permite tener distintos tipos de documentos en una misma colección.

- NPM [9]. Es el gestor de paquetes de JavaScript de Node que permite instalar dependencias sencillas, administrar módulos y distribuír paquetes.

- N-API [10]. Es un API para implementar complementos nativos. Es independiente del tiempo de ejecución de JavaScript y se mantiene como parte de Node. Las API expuestas por N-API normalmente se utilizan para crear y manipular valores de JavaScript.
- C++ [11]. Es un lenguaje de programación orientado a objetos, extendido del lenguaje de programación C.
- GraCT [1] [12]. Es una estructura compacta para almacenar trayectorias libres de objetos en movimiento, permitiendo consultas espacio-temporales. Almacena una cantidad ingente de datos utilizando el mínimo espacio y proporcionando un acceso rápido y eficiente para consultar datos.

Capítulo 3

Metodología y planificación

3.1 Metodología de desarrollo

La metodología que se decidió utilizar para llevar a cabo el desarrollo del proyecto fue una metodología iterativa incremental. Este método de desarrollo permite que el proyecto se pueda dividir en diferentes bloques, en los cuales el proceso de trabajo realizado es similar. Cada uno de los bloques es conocido como iteración. Esto nos facilita la posibilidad de, una vez finalizada la iteración, que el cliente evalúe el trabajo que se ha hecho y se planifica el siguiente incremento a abordar. En caso de considerarlo necesario, se pueden añadir nuevas funcionalidades o refinar las versiones anteriores. Esto es conocido como desarrollo iterativo incremental, donde el cliente obtendrá el producto final, que será el resultado de la implementación de las funcionalidades de todos los bloques realizados.

Al seguir un desarrollo incremental, el software debe ser construido de tal forma que se puedan añadir nuevas funcionalidades, posteriormente, de manera sencilla.

Esta metodología se ha inspirado en las mejores prácticas de Scrum. Scrum es un marco de trabajo que se utiliza para el desarrollo de proyectos software en equipo. En este proyecto no se ha aplicado Scrum como tal, ya que para ello es necesario un equipo de desarrollo que no había, solo se contaba con la alumna como desarrolladora. A continuación, se definen brevemente los roles, eventos y artefactos de Scrum [13] y se expone una breve explicación de cuáles se han utilizado en el desarrollo del proyecto y cómo se han utilizado.

3.1.1 Roles de Scrum

En Scrum existen tres roles importantes: Product Owner o Dueño de producto, Scrum Máster y Equipo de desarrollo.

- **Product Owner o Dueño de producto**: Es la persona que se encarga de mantener la comunicación con el cliente y que se lleve a cabo una buena gestión del Product Backlog,

así como priorizar las historias de usuario de este. Además solo puede existir uno por equipo Scrum y puede formar parte del equipo de desarrollo.

- **Scrum Máster**: Es la persona que más conocimientos tiene acerca de Scrum y es el responsable de que las técnicas de Scrum sean aplicadas en el equipo.
- **Equipo de desarrollo**: Son las personas que se encargan del desarrollo de las tareas indicadas por el Product Owner.

En este caso, el rol de dueño de producto y de equipo de desarrollo ha sido realizado por una misma persona, la alumna, aunque también se han tomado decisiones conjuntas con los directores del proyecto para priorizar las historias de usuario del Product Backlog que había que abordar en cada iteración.

3.1.2 Eventos de Scrum

A continuación, se describen los eventos de Scrum que se han utilizado en el desarrollo del proyecto junto con una pequeña explicación de cómo se han utilizado.

- **Sprint**: es la base de Scrum. Un sprint es un periodo corto de tiempo en el que se llevan a cabo las funcionalidades acordadas. Cada sprint comienza solamente cuando el anterior ya ha finalizado.
- **Reunión de planificación del sprint**: en esta reunión se acuerda entre todo el equipo cuales son las funcionalidades del trabajo que se va a llevar a cabo durante el siguiente sprint.
- **Revisión del sprint**: es una valoración que se lleva a cabo un vez finalizado el sprint y en donde se inspecciona el incremento de producto y se comprueban los resultados obtenidos.
- **Retrospectiva de sprint**: es una reunión en la que se analiza esa fase del proceso y de la cual se obtendrán conclusiones y mejoras para el siguiente sprint.

En este proyecto se han realizado seis sprints y en cada uno de ellos se han hecho las reuniones de planificación pertinentes para acordar, entre alumna y directores, su alcance. Al finalizar cada sprint, se hizo una revisión de este para ver si se había alcanzado el objetivo acordado. En el primer sprint, se decidió el alcance del proyecto y se hizo un estudio de las tecnologías que se iban a utilizar. En el segundo, se creó una lista con los requisitos del proyecto y se hizo una planificación de las funcionalidades que había que abordar. En los tres siguientes sprints se desarrolló el objetivo que se había acordado en cada una de las reuniones de planificación que se hizo anteriormente. En el último sprint se hizo una revisión completa del proyecto y se reservó un período de tres semanas para la elaboración de la memoria.

3.1.3 Artefactos de Scrum

- **Product Backlog**: es una lista con los requisitos del proyecto, conocida como historias de usuario que puede evolucionar a medida que se desarrolla el proyecto.
- **Sprint Backlog**: lista de trabajos que tiene que hacer el equipo durante el sprint para alcanzar el objetivo acordado.
- **Incremento**: es el resultado final de cada sprint. Forma en la que se mide cual ha sido el progreso del sprint.

Para este proyecto se han utilizado los tres artefactos descritos anteriormente. En primer lugar, la alumna ha creado una lista con las historias de usuario del proyecto. Posteriormente, se ha llegado a un acuerdo entre alumna y directores de cuales eran las historias de usuario que había que llevar a cabo en el siguiente sprint que se iba a desarrollar y se ha observado un progreso en cada uno de los sprints con el que finalmente se obtiene el resultado final.

3.1.4 Herramientas de apoyo a la metodología

- Balsamiq [14]. Herramienta para el desarrollo de prototipos de proyectos. Utilizada para el diseño de los mockups de la aplicación.
- Visual Studio Code [15]. Es un editor de código utilizado tanto para el desarrollo del cliente con Vue.js, como para el del servicio con Express.js, así como también para la integración del servicio con la estructura compacta GraCT.
- GitLab [16]. Es un servicio web para el control de versiones y de desarrollo software de equipos colaborativos. Está basado en Git. Este servicio permite gestionar los *sprints* mediante *milestones*, ya que de este modo se crean agrupaciones de las historias de usuario, y estas también se pueden gestionar creando *issues*, que son ramas locales en las cuales se lleva a cabo el desarrollo de una funcionalidad concreta.
- Workbench [17]. Es una herramienta viual que permite el desarrollo gráfico de bases de datos.
- Postman [18]. Es una herramienta que permite probar APIs de manera sencilla realizando peticiones sobre ellas.
- Overleaf [19]. Es un editor de LaTeX online, utilizado para el desarrollo de la memoria del proyecto.
- MagicDraw [20] y draw.io [21]. Son herramientas que permiten el diseño y creación de diagramas.

- Dia [22]. Es una herramienta que permite crear distintos tipos de diagramas. Se utilizó para la creación del modelo conceptual de datos de la aplicación.

3.2 Planificación y seguimiento

En esta sección se explica la planificación inicial que se ha hecho del proyecto y los recursos que se utilizaron durante el desarrollo de este y su seguimiento.

3.2.1 Planificación

El punto de partida de proyecto es fijar las bases del mismo, planificar las actividades a realizar con el fin de obtener determinados objetivos y resultados.

- Se hizo una primera reunión entre alumna y directores para hacer un análisis del proyecto y llegar a un acuerdo entre los miembros que forman parte y definir cual sería su alcance y la fecha de fin estimada. Además también se hizo un estudio de las tecnologías que se iban a utilizar en el desarrollo del proyecto.
- Una vez definido el alcance del proyecto, se determinaron los requisitos a partir de los cuales se crearon las historias de usuario. Y posteriormente, se hizo un prototipo de diseño de las pantallas de la aplicación que dan soporte a las historias de usuario definidas y se creó el modelo de datos adecuado.
- Se dividió el tiempo estimado para la realización del proyecto en cuatro iteraciones de aproximadamente un período de tres semanas cada iteración. En cada iteración se implementarían un conjunto de historias de usuario agrupadas de manera que tuvieran una complejidad similar.
- Finalmente, se reserva un período de tres semanas para la elaboración de la memoria del proyecto.

Se pueden destacar dos tipos de recursos utilizados en este proyecto que se explican a continuación brevemente:

- Recursos humanos: personas adecuadas y capacitadas para realizar las tareas necesarias para el cumplimiento de los objetivos del proyecto. El personal debe tener las cualificaciones requeridas para las funciones que hay que realizar y se debería indicar quién es responsable de lo que se desarrolla en el proyecto y cómo se distribuye el trabajo. En este proyecto formaron parte cuatro personas cualificadas, tres de ellos los directores de este, entre los que se encuentran quienes implementaron la estructura compacta GraCT para la cuál se propuso el desarrollo del visor cartográfico. Y la alumna, encargada del análisis, diseño, implementación y pruebas del proyecto.

- Recursos técnicos: herramientas, equipos, instrumentos y tecnologías que se necesitan para llevar a cabo el proyecto. Los recursos software necesarios para el desarrollo de este fueron todas las herramientas descritas en Apartado 3.1.4, las cuales no supusieron ningún coste adicional ya que se tratan de herramientas de código abierto. Y los recursos hardware que se usaron fue únicamente el portátil de la alumna.

3.2.2 Seguimiento

Se hizo un seguimiento de la planificación descrita en el Apartado 3.2.1 para comprobar que las acciones se ejecutaban de manera correcta. A continuación, se detallan los sprints que se llevaron a cabo durante el desarrollo y lo qué se ha hecho en cada uno de ellos.

- Determinación de alcance: se hizo una primera reunión en la cual se acordó el alcance del proyecto y en ella se realizó un análisis preliminar de los requisitos para proceder al diseño inicial de la aplicación web. Durante este primera iteración se hizo un estudio tecnológico de las mejores opciones tecnológicas que se podían aplicar en el desarrollo del proyecto.

- Análisis preliminar: en la segunda reunión se tomaron decisiones para llegar a un acuerdo de las mejoras que se podían hacer en algunos de los mockups diseñados como prototipo inicial, priorizar las funcionalidades y agruparlas en conjuntos con complejidad similar. Se destacan cuatro grupos. Por una parte la gestión de usuarios, barcos y grupos. Por otra parte, la integración con la estructura compacta GraCT, implementación de las consultas a dicha estructura y por último, la implementación de los reportes que el usuario puede hacer sobre los barcos.

- Sprint 1: una vez hecha la agrupación de las historias de usuario que se iban a desarrollar, se llevó a cabo la implementación y pruebas del primer conjunto, que serían la gestión de usuarios, barcos y grupos.

- Sprint 2: en esta iteración se hizo la integración de la aplicación cliente con la estructura compacta de GraCT para que fuese posible la comunicación entre front-end y back-end.

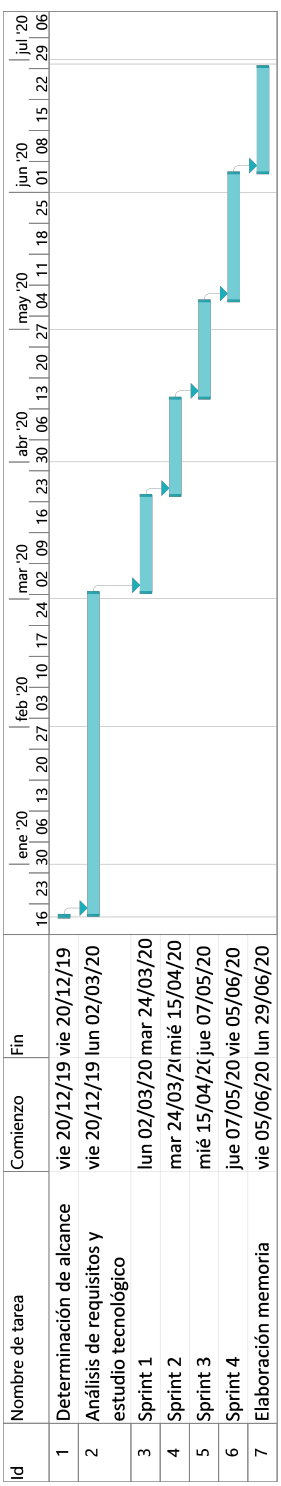
- Sprint 3: una vez conseguida la integración con GraCT se implementó la librería que permite hacer las consultas contra dicha estructura y además se hicieron las correspondientes pruebas para comprobar que las operaciones funcionaban de manera adecuada.

- Sprint 4: finalizada la implementación de las operaciones, se mejoró la interfaz de usuario de la aplicación y se hicieron optimizaciones en las consultas a la estructura compacta.

- Sprint 5: elaboración de la memoria.

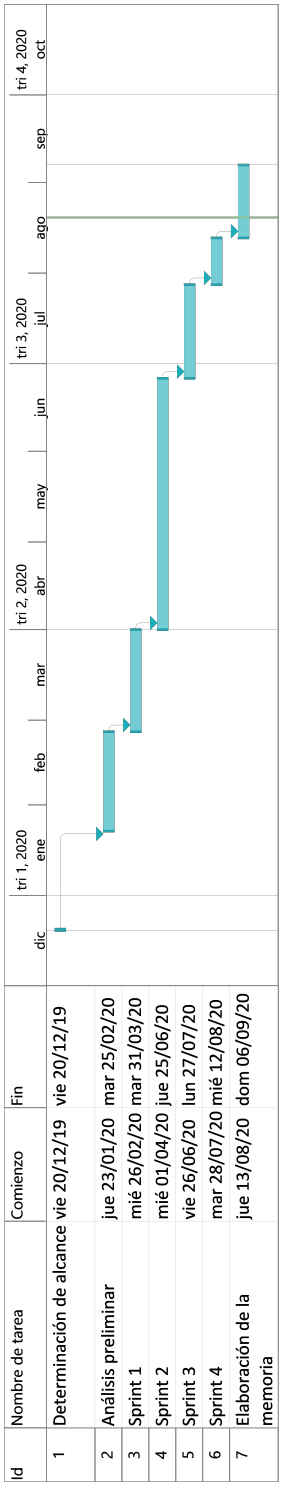
Como se puede ver en el diagrama de Gantt de la Figura 3.1b, la planificación inicial del proyecto no se cumplió. Durante el desarrollo del proyecto se vivió una situación especial de confinamiento por la pandemia COVID-19 que afectó al desarrollo normal del proyecto por parte de la alumna. Durante este período se facilitó la comunicación con los tutores mediante reuniones realizadas a través de la herramienta Teams, ya que gran parte del proyecto se tuvo que llevar en la distancia.

En la Figura 3.1, se muestra un diagrama de Gantt de la planificación del proyecto y otro del seguimiento, donde se puede comparar el tiempo estimado con el tiempo real de la duración del proyecto.



Id	Nombre de tarea	Comienzo	Fin
1	Determinación de alcance	vie 20/12/19	vie 20/12/19
2	Análisis de requisitos y estudio tecnológico	vie 20/12/19	lun 02/03/20
3	Sprint 1	lun 02/03/20	mar 24/03/20
4	Sprint 2	mar 24/03/2(	mié 15/04/20
5	Sprint 3	mié 15/04/2(	jue 07/05/20
6	Sprint 4	jue 07/05/20	vie 05/06/20
7	Elaboración memoria	vie 05/06/20	lun 29/06/20

(a) Diagrama de Gantt de la planificación



Id	Nombre de tarea	Comienzo	Fin
1	Determinación de alcance	vie 20/12/19	vie 20/12/19
2	Análisis preliminar	jue 23/01/20	mar 25/02/20
3	Sprint 1	mié 26/02/20	mar 31/03/20
4	Sprint 2	mié 01/04/20	jue 25/06/20
5	Sprint 3	vie 26/06/20	lun 27/07/20
6	Sprint 4	mar 28/07/20	mié 12/08/20
7	Elaboración de la memoria	jue 13/08/20	dom 06/09/20

(b) Diagrama de Gantt del seguimiento

Figura 3.1: Diagramas de Gantt de la planificación y seguimiento del proyecto.

Capítulo 4
Análisis

4.1 Requisitos

Esta aplicación se basa en la gestión de flota, para la cual se desarrolló un visor de mapas que permite al usuario acceder a diferentes funcionalidades descritas en el Apartado 4.1.2, y que permite interactuar con estructuras de compresión, en este caso, con la estructura GraCT, de manera sencilla.

4.1.1 Actores

A continuación, se detallan los roles de usuario que existen en esta aplicación, citando las funcionalidades a las que tiene acceso cada uno de ellos.

- **Usuario registrado.** Este usuario tiene que estar registrado en el sistema y puede realizar consultas de los movimientos y localizaciones de los barcos a los que él tiene acceso, que serían aquellos que pertenecen a sus grupos. También puede acceder a la diferente información de los barcos que son visibles para él, así como informar de posibles problemas o información de interés sobre estos barcos. Todo usuario registrado tendrá acceso a su información personal, la cual puede ser editada en cualquier momento.
- **Administrador.** Además de las consultas a la estructura GraCT permitidas para todo usuario registrado, también se encarga de la gestión de los usuarios, barcos y grupos que existen en el sistema.

4.1.2 Requisitos funcionales

Los requisitos funcionales son aquellos que describen los servicios que el sistema prestará al usuario y la forma en la que el sistema reaccionará a cada uno de estos servicios. En el Cuadro 4.1, se recogen las historias de usuario de la aplicación.

ID	Nombre	Descripción
HU-01	Autenticación	Proceso de autenticación en el sistema por parte del usuario.
HU-02	Consultar ruta (a)	Realizar una petición para consultar la última posición de los barcos.
HU-03	Consultar ruta (b)	Realizar una petición para consultar la ruta de un barco concreto.
HU-04	Añadir report	Crear un nuevo informe de un barco.
HU-05	Ver report	Ver los reports de los barcos que se están mostrando en el mapa.
HU-06	Consultar región	Realizar una petición para consultar los barcos que están en una región concreta.
HU-07	Consultar trayectoria	Realizar una petición para consultar la trayectoria de todos los barcos que están en una región concreta.
HU-08	Ver trayectoria	Ver la trayectoria de un barco concreto del resultado obtenido al consultar la trayectoria(HU-07).
HU-09	Consultar punto	Realizar una petición para consultar los barcos más próximos a un punto indicado y en un instante de tiempo concreto.
HU-10	Ver mi perfil	Ver el perfil del usuario autenticado.
HU-11	Editar mi perfil	Editar el perfil del usuario autenticado.
HU-12	Cambiar contraseña	Modificar la contraseña del usuario autenticado.
HU-13	Búsqueda avanzada	Realizar una petición para buscar los barcos a partir de los datos indicados.
HU-14	Mostrar resultado en mapas	Visualizar en el mapa el resultado obtenido de la búsqueda avanzada (HU-13) realizada.
HU-15	Ver última posición	Mostrar en el mapa la última posición de un barco concreto.
HU-16	Ver usuarios	Ver la lista de usuarios registrados en el sistema. Solo la pueden ver aquellos usuarios con permisos de administrador.
HU-17	Ver usuario	Ver la información de un usuario concreto. Solo lo pueden ver aquellos usuarios con permisos de administrador.
HU-18	Editar usuario	Editar la información de un usuario, por parte del usuario administrador.
HU-19	Añadir usuario	Registrar un nuevo usuario. Solo lo puede registrar un usuario con permisos de administrador.
HU-20	Eliminar usuario	Eliminar un usuario registrado. Solo lo puede eliminar un usuario con permisos de administrador.
HU-21	Dar de alta usuario	Dar de alta un usuario que estaba dado de baja. Solo lo puede dar de alta un usuario con permisos administrador.

HU-22	Asignar grupo	Asignarle un grupo a un usuario o a un barco. Solo lo puede asignar un usuario con permisos de administrador.
HU-23	Ver barcos	Ver la lista de los barcos registrados en el sistema. Solo la pueden ver aquellos usuarios con permisos de administrador.
HU-24	Añadir barco	Registrar un nuevo barco en el sistema. Solo lo puede registrar un usuario con permisos de administrador.
HU-25	Eliminar barco	Eliminar un barco registrado. Solo lo puede registrar un usuario con permisos de administrador.
HU-26	Ver barco	Ver la información de un barco concreto.
HU-27	Editar barco	Editar los datos de un barco concreto. Solo lo puede editar un usuario con permisos de administrador.
HU-28	Ver grupos	Ver la lista de los grupos registrados en el sistema. Solo la pueden ver usuarios con permisos de administrador.
HU-29	Eliminar grupo	Eliminar un grupo registrado. Solo lo puede eliminar un usuario con permisos de administrador.
HU-30	Ver grupo	Ver la información de un grupo concreto.
HU-31	Crear grupo	Registrar un nuevo grupo en el sistema. Solo lo puede crear un usuario con permisos de administrador.
HU-32	Asignar usuario	Asignar un usuario a grupo. Solo lo puede asignar un usuario con permisos de administrador.
HU-33	Quitar usuario	Eliminar un usuario de un grupo concreto. Solo lo puede quitar un usuario con permisos de administrador.
HU-34	Asignar barco	Asignar un barco a un grupo. Solo lo puede asignar un usuario con permisos de administrador.
HU-35	Quitar barco	Eliminar un barco de un grupo concreto. Solo lo puede quitar un usuario con permisos de administrador.
HU-36	Cerrar sesión	Proceso de cierre de sesión en el sistema por parte del usuario.

Cuadro 4.1: Tabla de Product Backlog.

A continuación, se explican detalladamente cada una de las historias de usuario que se recogen en el Cuadro 4.1.

- **HU-01 Autenticase.** El usuario introduce sus credenciales para acceder en la aplicación. El sistema valida las credenciales y en caso de que sean válidas el usuario estará autenticado y tendrá acceso a todas las funcionalidades disponibles. En caso contrario, se mostrará un mensaje de error.

- **HU-02 Consultar ruta (a).** Una vez autenticado el usuario, se visualizará, en la pantalla principal de la aplicación, la última posición de los barcos. Esta posición está almacenada en base de datos ya que previamente se hace una petición contra la estructura compacta y se guarda el resultado obtenido en la base de datos. También se muestra una tabla resultado con los barcos que se están visualizando en el mapa y para cada uno podrá acceder a su ficha correspondiente, ver la última posición del barco concreto y hacer algún comentario de este. Se pueden ver también dos tablas, una con los reports que hizo el propio usuario sobre los barcos y una segunda con los reports de los barcos que pertenecen a su grupo.

- **HU-03 Consultar ruta (b).** Petición que hace el usuario para obtener la trayectoria de un barco, indicando el nombre de este y el intervalo de tiempo que desea consultar. Una vez hecha la petición se visualizará en el mapa la trayectoria del barco en caso de que exista o un mensaje informativo indicando que el barco no ha hecho ninguna trayectoria en el intervalo de tiempo indicado.

- **HU-04 Añadir report.** Permite al usuario crear informes de los barcos que son visibles para él.

- **HU-05 Ver report.** Una vez añadido el informe, todos los usuarios que estén asignados a grupos a los que pertenezca el barco, podrán ver estos informes. En la pantalla principal se visualizará una tabla con los reports de los barcos que se están mostrando en el mapa.

- **HU-06 Consultar región.** Desde el menú lateral, el usuario accede a esta funcionalidad que le permite hacer una petición para consultar los barcos que están en una región concreta, en un instante de tiempo indicado por el usuario. La región la selecciona el propio usuario en el mapa, que se muestra en pantalla, pulsando control *(ctrl)* y el área deseada. Se visualizará en el mapa la región marcada con los barcos que cumplen las condiciones y además se mostrará una tabla con los barcos resultado de la consulta, de los cuales puede ver su ficha y su última posición.

- **HU-07 Consultar trayectoria.** En la misma pantalla, en donde se ejecuta la historia de usuario HU-06 (consultar región), el usuario puede indicar un intervalo de tiempo y seleccionar el área sobre la cuál desea hacer la búsqueda y se visualizarán en el mapa la trayectoria de todos los barcos que hayan pasado por esa región en este intervalo de tiempo, junto con una tabla con los barcos resultados de la búsqueda. Tendrá acceso a la ficha de cada barco y podrá consultar la trayectoria de un barco concreto.

- **HU-08 Ver trayectoria.** Una vez realizada la búsqueda de los barcos de una región en

un intervalo de tiempo, el usuario podrá ejecutar esta funcionalidad para cada uno de los barcos, para ver la ruta concreta.

- **HU-09 Consultar punto.** Desde el menú lateral, el usuario puede hacer una búsqueda de un número determinado de barcos que están próximos a un punto de coordenadas, seleccionado en el mapa, en un instante de tiempo indicado por el usuario. Se visualizará en el mapa la última posición de los barcos y una tabla con el resultado obtenido, donde se podrá consultar la ficha y la última posición de cada barco.

- **HU-10 Ver mi perfil.** El usuario autenticado puede ver su información personal almacenada en el sistema. Tendrá acceso a la historia de usuario HU-11 (editar mi perfil).

- **HU-11 Editar mi perfil.** El usuario puede modificar/actualizar su información personal que está almacenada en base de datos.

- **HU-12 Cambiar contraseña.** Proceso mediante el cual el usuario puede modificar su contraseña actual.

- **HU-13 Búsqueda avanzada.** Permite al usuario realizar una consulta para buscar los barcos en función de los atributos deseados. El resultado se mostrará en una tabla y podrá visualizarlo en el mapa.

- **HU-14 Mostrar en mapas.** Visualizar en el mapa la última posición de los barcos resultado de la ejecución de la historia de usuario HU-13 (búsqueda avanzada).

- **HU-15 Ver última posición.** Funcionalidad que permite al usuario visualizar en el mapa la última posición registrada que existe de un barco concreto.

- **HU-16 Ver usuarios.** El administrador tiene acceso, desde el menú lateral, a una tabla con los usuarios registrados en el sistema que no están dados de baja. Para cada uno de ellos se muestra su login, que es único, y tiene disponibles iconos para ejecutar las historias de usuario que se mencionan: un icono con forma de ojo para ejecutar a HU-17 (ver usuario), un icono con forma de lápiz para acceder a HU-18 (editar usuario) y un icono con forma de papelera para acceder a HU-20 (eliminar usuario). Como los usuarios se pueden dar de baja, se muestra, en la misma página, una tabla con los usuarios que están dados de baja en el sistema y para cada uno tendrá un icono para acceder a HU-21 (dar de alta usuario) y un icono con forma de ojo para acceder a HU-17 (ver usuario). Para facilitar la búsqueda de un usuario, en la propia tabla, existe un buscador que filtra los usuarios por login. Puede acceder de forma rápida a través de un botón disponible en esta pantalla a la historia de usuario HU-19 (añadir usuario).

- **HU-17 Ver usuario.** Esta historia de usuario le permite al administrador consultar la información personal de un usuario concreto registrado en el sistema y también muestra en la misma página una tabla con todos los grupos a los que pertenece, facilitando el acceso a la historia de usuario HU-34 (quitar usuario) a través de un icono. En la misma pantalla dispone de un botón que le proporciona acceso a la historia de usuario HU-22 (asignar grupo), para ingresar al usuario en un nuevo grupo.
- **HU-18 Editar usuario.** A partir de la historia de usuario HU-17 (ver usuario) o de la historia de usuario HU-16 (ver usuarios), el administrador puede modificar/actualizar la información personal de un usuario concreto.
- **HU-19 Añadir usuario.** Es el proceso mediante el cual el administrador registra a un nuevo usuario en el sistema. Por defecto, todos los usuarios registrados se crean sin permisos especiales. Tiene acceso desde la historia de usuario HU-16 (ver usuarios).
- **HU-20 Eliminar usuario.** Es el proceso que lleva a cabo el administrador para dar de baja a un usuario en el sistema, una vez dado de baja un usuario, se dará de baja de todos los grupos a los que esté asignado y no tendrá acceso a la aplicación hasta que no sea dado de alta de nuevo. Tiene acceso a esta funcionalidad desde la historia de usuario HU-16 (ver usuarios).
- **HU-21 Asignar grupo.** A partir de la historia de usuario HU-17 (ver usuario), el administrador lleva a cabo este proceso para añadir al usuario concreto en un nuevo grupo.
- **HU-23 Ver barcos.** El administrador tiene acceso, desde el menú lateral, a una tabla con todos los barcos registrados en el sistema. Para cada uno de ellos se muestra el nombre junto con el MMSI (identificador único del barco) y puede ejecutar, a través de un icono con forma de lápiz, la historia de usuario HU-27 (editar barco), un icono con forma de ojo para ver la ficha de un barco (HU-26 (ver barco)) y un icono con forma de papelera para eliminar un barco (HU-25 (eliminar barco)). Además, para facilitar la búsqueda de un barco concreto, como la cantidad de estos puede ser grande, la tabla cuenta con un buscador que filtra por nombre o identificador del barco. También dispone de un botón para ejecutar la historia de usuario HU-24 (añadir barco) en la misma página.
- **HU-24 Añadir barco.** El administrador puede ejecutar esta funcionalidad desde la historia de usuario HU-23 (ver barcos), en donde se cubre un formulario para registrar un nuevo barco en el sistema. Para registrarlo solo es obligatorio el MMSI, que es un número de 8 caracteres que identifica al barco, los datos restantes son opcionales. Una vez registrado el barco, se visualiza en pantalla el listado de todos los barcos, a los cuales solo tiene acceso el administrador, para el resto de usuarios registrados solo serán visibles los que pertenezcan a sus mismos grupos.

- **HU-25 Eliminar barco.** Es el proceso mediante el cual el administrador da de baja un barco del sistema. En consecuencia, el barco será dado de baja de todos los grupos a los que esté asignado. Se accede a través de la historia de usuario HU-23 (ver barcos).
- **HU-26 Ver barco.** A partir de la historia de usuario HU-23 (ver barcos), el administrador puede consultar toda la información almacenada de un barco y desde la misma página, puede editar el barco (HU-27 (editar barco)) y ver los grupos a los que pertenece, así como también ejecutar la funcionalidad que le permite asignarlo a un grupo (HU-22 (asignar grupo)). La información del barco solo será visible para aquellos usuarios que pertenezcan a los grupos a los que esté asignado el barco.
- **HU-27 Editar barco.** Esta historia de usuario le permite al administrador modificar/actualizar la información del barco. Puede ejecutar esta funcionalidad a través de las historias de usuario HU-23 (ver barcos) y HU-26 (ver barco).
- **HU-28 Ver grupos.** Desde el menú lateral, el administrador puede acceder a un listado con los grupos que están registrados en el sistema. Para cada grupo se visualiza su nombre y se pueden ejecutar desde esta pantalla las funcionalidades de eliminar grupo (HU-29 (eliminar grupo)) a través de un icono con forma de papelera y ver la ficha del grupo, (HU-30 (ver grupo)), a través de un icono con forma de ojo. Además, tiene disponible un botón para ejecutar la historia de usuario HU-31 (crear grupo), para añadir un nuevo grupo.
- **HU-29 Eliminar grupo.** Proceso que le permite al administrador dar de baja un grupo en el sistema. Como consecuencia, todos los usuarios y barcos asignados a dicho grupo serán dados de baja en el mismo.
- **HU-30 Ver grupo.** A partir de la historia de usuario HU-28 (ver grupos), el administrador puede consultar la ficha de un grupo, en donde se muestra una tabla con los usuarios que pertenecen a este grupo y un icono para cada uno de ellos que le permite al administrador ejecutar la historia de usuario HU-33 (quitar usuario) y otra tabla donde se muestran los barcos asignados al grupo con un icono para cada uno que le permitirá al administrador quitar un barco del grupo (HU-35 (quitar barco)). Desde esta ficha, puede añadir más usuarios a través de un botón que ejecutará la historia de usuario HU-32 (asignar usuario), así como también añadir más barcos al grupo mediante un botón que ejecutará la historia de usuario HU-34 (asignar barco).
- **HU-31 Crear grupo.** Permite registrar un nuevo grupo en el sistema, cada grupo tendrá un nombre que será único. Una vez creado el nuevo grupo se pueden ejecutar las historias de usuario HU-32 (asignar usuario) y HU-34 (asignar barco), aunque no es obligatorio, estas operaciones pueden realizarse en cualquier otro momento.

- **HU-32 Asignar usuario.** Proceso mediante el cual el administrador añade un usuario a un grupo. Una vez que el usuario ha sido asignado a un grupo, este tendrá acceso a la información de los barcos que pertenezcan al mismo.
- **HU-33 Quitar usuario.** Proceso que le permite al administrador eliminar a un usuario de un grupo concreto. El mismo usuario puede volver a formar parte de este grupo.
- **HU-34 Asignar barco.** Proceso que permite al administrador añadir un barco en un grupo concreto. Este barco será visible para todos los miembros del grupo.
- **HU-35 Quitar barco.** Proceso a través del cual el administrador elimina un barco de un grupo concreto. El mismo barco puede volver a formar parte de este grupo.
- **HU-36 Cerrar sesión.** Proceso mediante el cual un usuario autenticado termina su sesión en el sistema.

4.1.3 Requisitos no funcionales

Los requisitos no funcionales definen las características que tiene el sistema, como son la capacidad de almacenamiento, la fiabilidad, la seguridad, etc. Estos requisitos son las propiedades o cualidades que debe tener el producto, no se refieren directamente al comportamiento que suministra el sistema.

Los requisitos no funcionales de este proyecto son:

- La aplicación debe permitir al usuario hacer consultas de manera intuitiva y proporcionarle un interfaz que sea fácil de utilizar.
- El acceso a la información almacenada en la estructura compacta debe ser eficiente, ya que en esta se guardan grandes cantidades de datos usando el mínimo espacio necesario.
- Tiempos de respuesta cortos al realizar peticiones contra la estructura de compresión utilizada.

4.2 Arquitectura del sistema

La arquitectura del sistema está basada en la arquitectura en 3 capas. Esta es una arquitectura lineal con una jerarquía rígida, de manera que solo se conoce la capa inmediatamente inferior. La comunicación entre ellas suele ser asíncrona para proporcionar mayor escalabilidad al sistema. Los componentes de esta arquitectura son los que se explican a continuación brevemente y se puede ver en la Figura 4.1 un diagrama de componentes de esta.

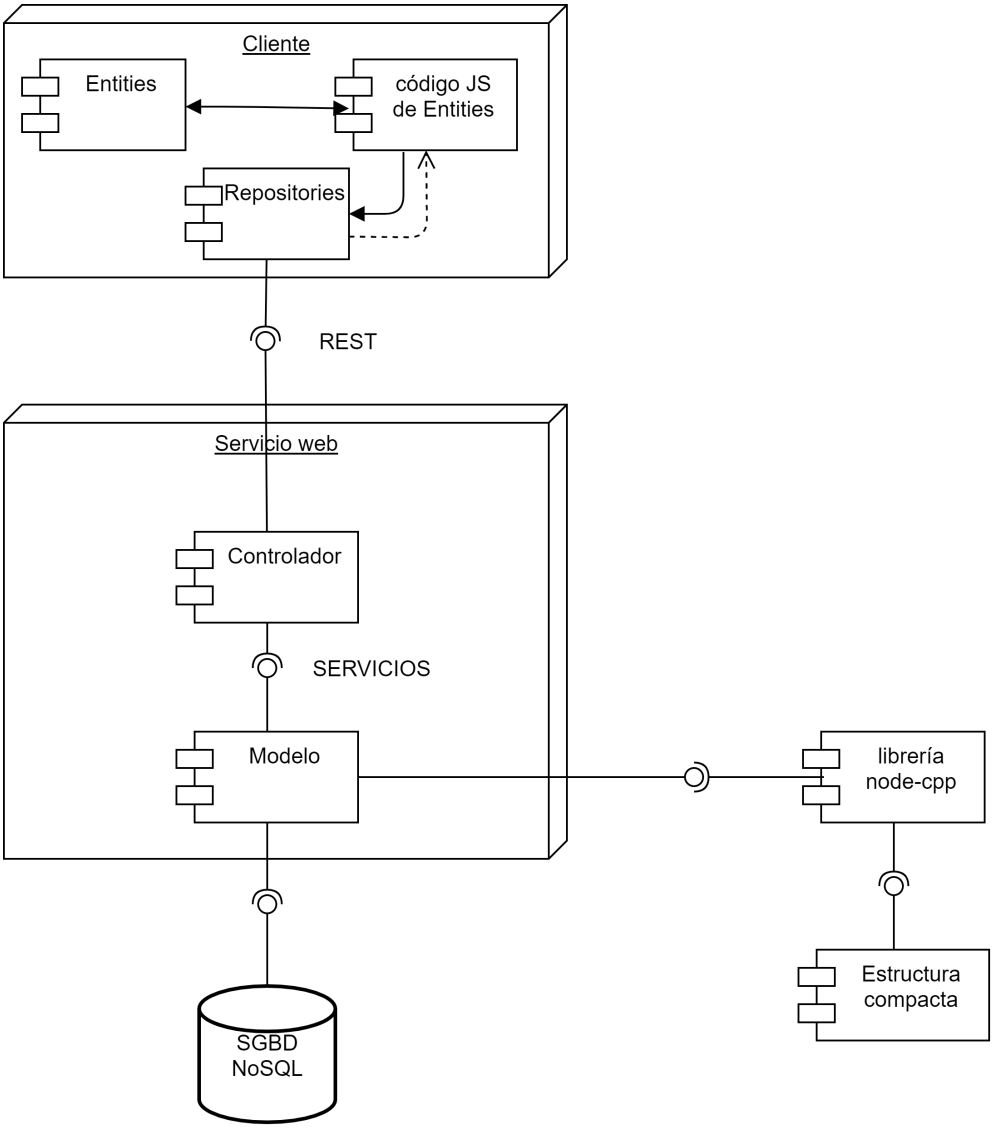


Figura 4.1: Diagrama de componentes genérico de la arquitectura del sistema.

- **Capa cliente.** En esta capa se crea la interfaz de usuario de la aplicación, interactúa con el usuario. Tiene como función recibir los datos introducidos por el usuario y enviarlos a la capa lógica de negocio, que es con la única con la que mantiene comunicación.

  Para el desarrollo de esta capa se utilizó el patrón arquitectónico *Model-View-ViewModel* (MVVM). Este patrón arquitectónico está compuesto por los componentes ***Model***, que se correspondería con el componente "Repositories" de nuestra capa cliente, que son los datos del dominio, que a su vez se comunican con el servicio REST; ***View***, que se corresponde con el componente "Entities" de nuestra capa cliente y es la interfaz de usuario; y ***ViewModel***, que ser correspondería con el componente "Código JS de Enti-

ties" de nuestra capa cliente, el cual contiene el código fuente JavaScript asociado a la vista y se utiliza para acceder a los métodos y propiedades públicas.

- **Capa lógica de negocio/controlador.** Implementa la lógica de negocio, utilizando la capa de accceso a datos para leer o escribir los datos que necesita. El controlador se comunica tanto con la capa cliente como con la capa de acceso a datos. De la capa cliente recibe los datos de las peticiones que hace el usuario, y la comunicación con la capa de acceso a datos le permite tanto enviar los datos para que se almacenen en el sistema de gestión de base de datos, como leerlos para realizar las operaciones necesarias para obtener el resultado de la petición.
- **Capa de acceso a datos/modelo.** Es la capa encargada de almacenar los datos del sistema. Tiene como función devolver y guardar los datos de la capa lógica de negocio. Tiene un sistema de gestor de base de datos al cual solo tiene acceso el controlador.
- **Librería node-cpp.** Es una librería que permite la comunicación entre el back-end de la aplicación y la estructura GraCT. Fue necesaria su creación debido a que el desarrollo del servicio se hizo en Node.js y la estructura compacta está diseñada en C++. Por lo tanto, había que crearla para poder integrar ambos sistemas.
- **Estructura compacta.** En esta aplicación se utiliza también una estructura compacta, GraCT, que permite almacenar grandes cantidades de datos en el mínimo espacio necesario y consultar de forma eficiente las trayectorias de los barcos y consultas más complejas. Esta estructura se conecta a través de una interfaz con el back-end de la aplicación.

4.3 Interfaz de usuario

En el análisis preliminar del proyecto se hizo un prototipo de diseño de la interfaz gráfica de la aplicación, que se detalla en esta sección incluyendo los mockups de este para visualizar cómo era el diseño inicial. Aquí solo se explican aquellas que se han considerado más relevantes porque debido a la falta de espacio no se han podido añadir todas, las restantes se pueden encontrar en Apéndice A. Cabe destacar que se han ido haciendo pequeñas modificaciones durante el desarrollo de la interfaz de usuario para facilitar al usuario la interacción con el sistema. Estas modificaciones son posibles ya que se utiliza una metodología incremental.

Una vez que el usuario se autenticó en el sistema, la pantalla principal de la aplicación sería la de la Figura 4.2, donde se visualizaría en el mapa la última posición de los barcos a los que tiene acceso el usuario, junto con una tabla resultado que le permite acceder a funcionalidades concretas de los barcos, así como una tabla con los informes que el propio usuario hizo de los barcos y los de los grupos a los que pertenece.

En el visor de mapas se pueden realizar diferentes tipos de consultas, entre ellas obtener la trayectoria de los barcos que pasaron por un área concreta en un intervalo de tiempo seleccionado. Se puede ver en la Figura 4.3 el prototipo de esta pantalla, en donde, para facilitar al usuario la visualización de los barcos resultado de la consulta realizada, se muestra una tabla con estos. En esta tabla, se puede acceder a la trayectoria concreta de cada uno de ellos, así como a su ficha, que contiene toda la información que hay almacenada en el sistema. La idea inicial era acceder a todas las trayectorias de los barcos a través del botón "TRAYECTORIA", que visualizaría en el mapa las rutas de los barcos como en la Figura 4.4, aunque finalmente, como se puede ver en la Figura 7.8a, esto fue modificado para que el usuario pudiera ver en el mapa directamente las trayectorias sin tener que hacerlo accediendo a través de un botón. El usuario tiene que ingresar las fechas de inicio y de fin para poder obtener la trayectoria y en caso de que estas fechas coincidan, se visualizará en el mapa la última posición de los barcos de la región. Además, también se podían ver los distintos informes de los barcos, tanto los propios del usuario como los de los grupos a los que pertenece. Sin embargo, en la aplicación final, esto solo se verá en la página principal.

Otra de las consultas que se puede realizar, es la búsqueda de los barcos más próximos a un punto, como se puede ver en la Figura 4.5, en donde el usuario indica el instante de tiempo y el número de barcos que quiere obtener. Del mismo modo que en la Figura 4.3, se mostrará una tabla resultado con todos los barcos obtenidos.

Otra de las funcionalidades destacable de esta aplicación, es que permite al usuario realizar búsquedas avanzadas de los barcos. El usuario puede realizar cualquier búsqueda indicando las características por las que desea filtrar los barcos. Esta búsqueda devolverá un resultado similar al de la Figura 4.6. Desde aquí se puede visualizar la última posición de cada barco concreto o visualizar la última localización de todos a través del botón "Mostrar en mapas".

Además de las funcionalidades descritas anteriormente, el usuario administrador se encarga de la gestión de usuarios, barcos y grupos. Como el prototipo de estas pantallas es similar en los tres casos se muestran solo algunas de ellas. Tanto para los usuarios, como barcos y grupos, el administrador tendrá acceso a una lista como la de la Figura 4.7, a partir de la cual podrá acceder a la información concreta del elemento listado y editarlo, así como darlo de baja o registrar un nuevo elemento en el sistema cubriendo todos los datos necesarios en un formulario.

Un usuario registrado, tendrá acceso a su información personal almacenada en el sistema, la cual podrá modificar/actualizar siempre y cuando lo desee.

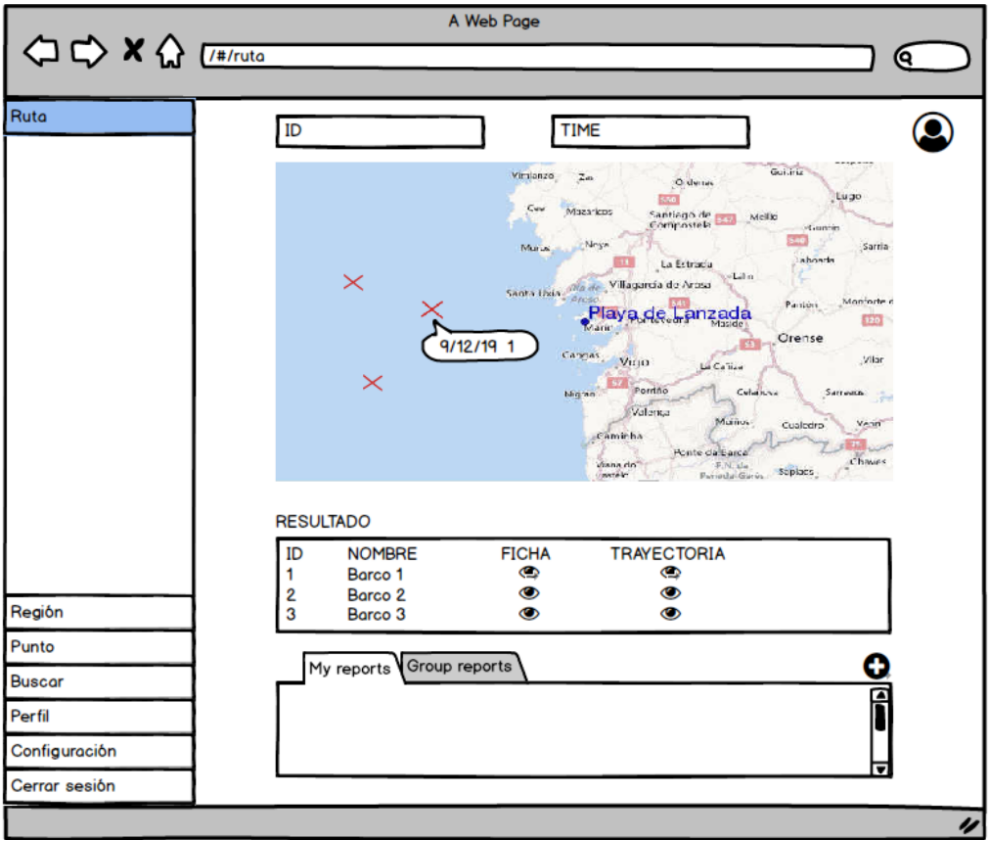


Figura 4.2: Mockup de la pantalla principal de la aplicación.

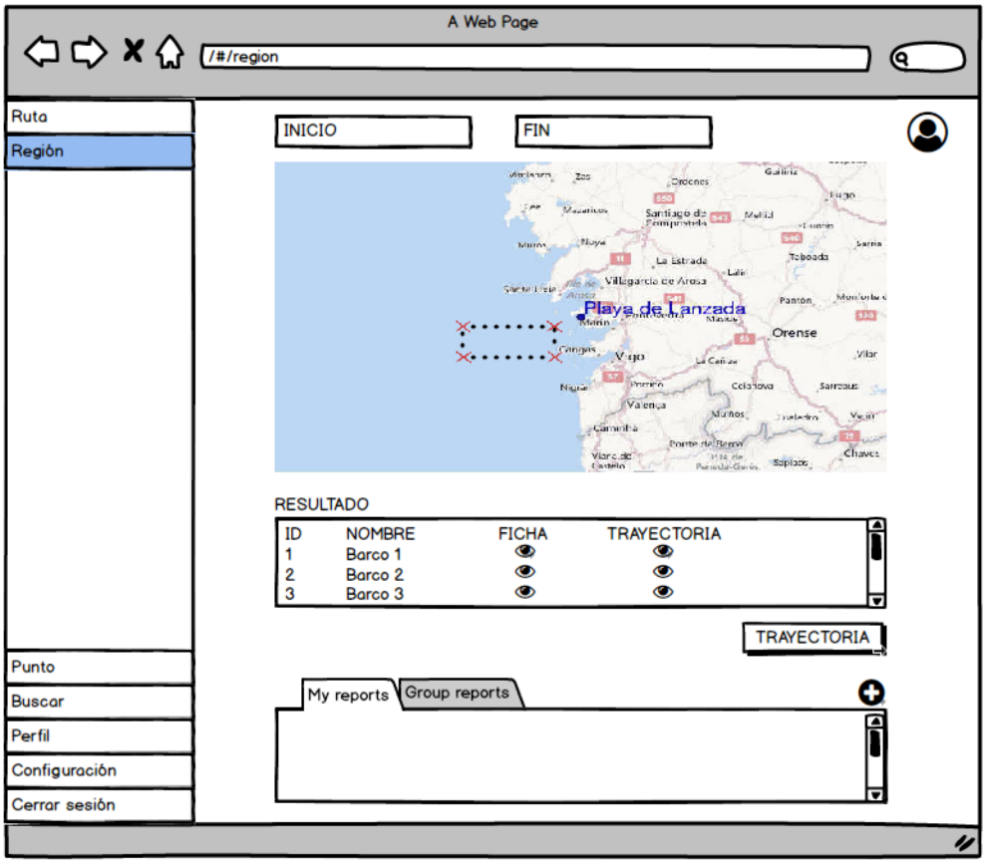


Figura 4.3: Mockup de la consulta de los barcos en una región.

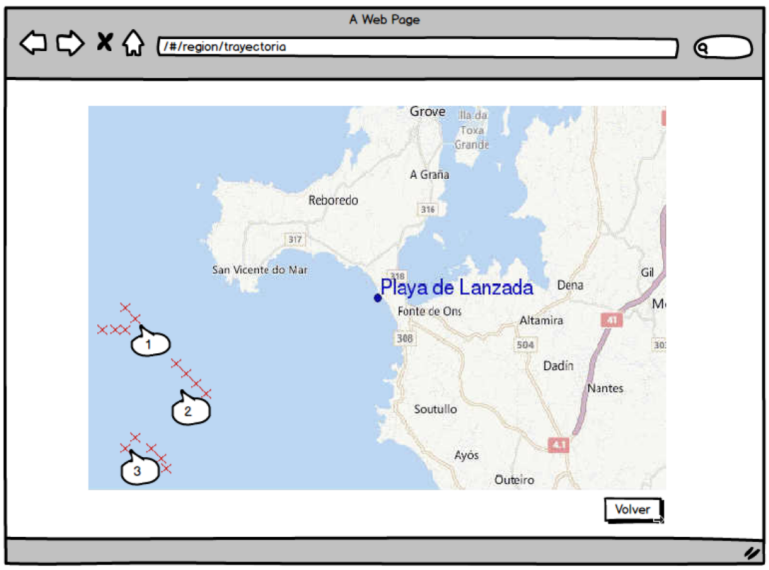


Figura 4.4: Mockup de las trayectorias de los barcos de una region en un intervalo de tiempo.

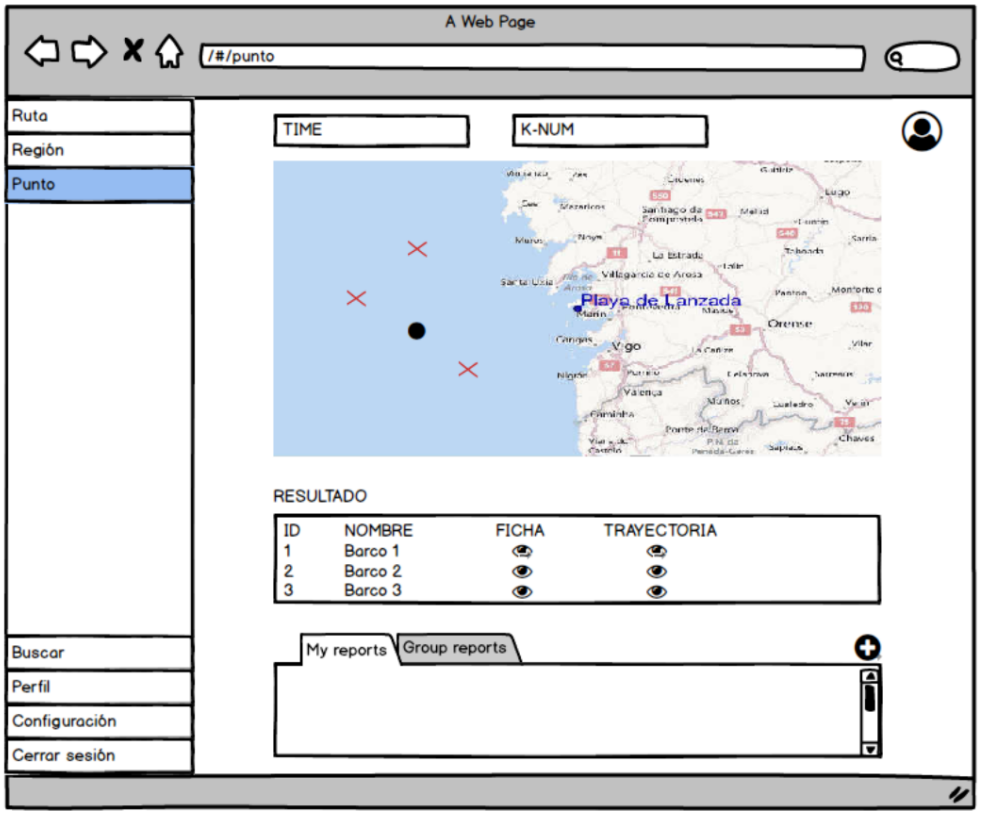


Figura 4.5: Mockup de la consulta de los n barcos más próximos a un punto.

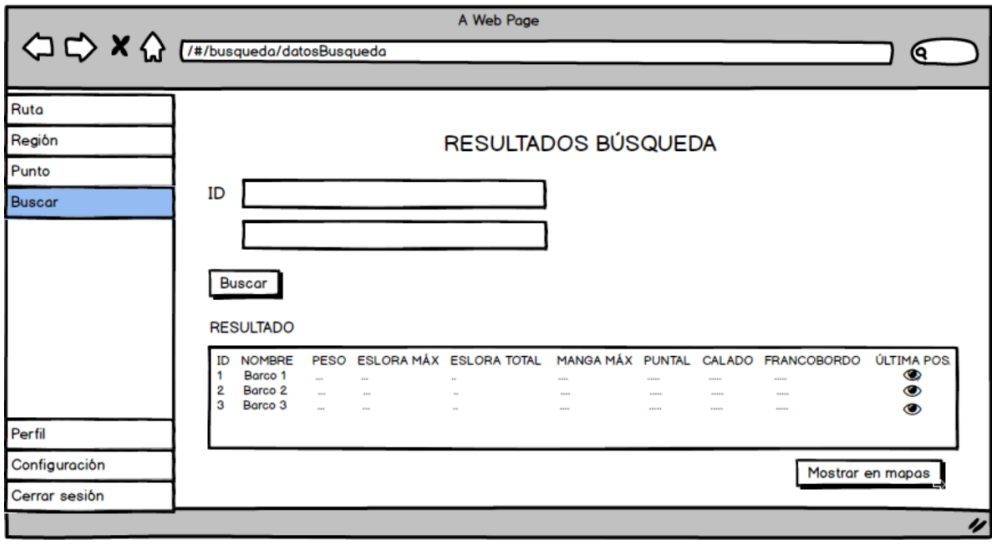


Figura 4.6: Mockup del resultado obtenido de la búsqueda avanzada realizada por un usuario.

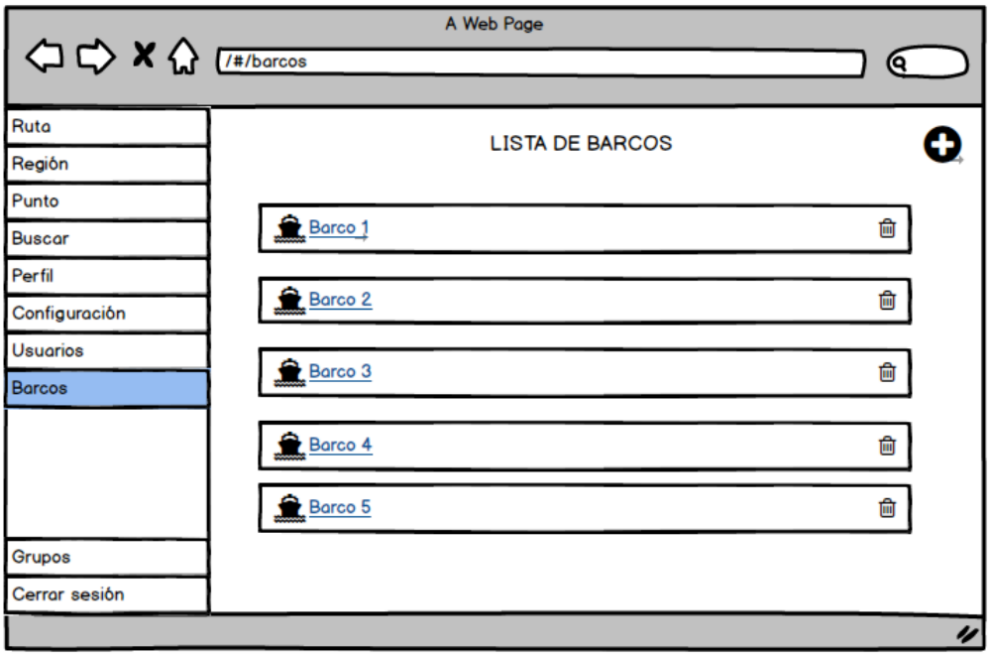


Figura 4.7: Mockup del listado de todos los barcos del sistema.

4.4 Modelo conceptual de datos

En esta sección se hace una distinción entre los datos almacenados en la base de datos MongoDB, que es la base de datos conceptual de la aplicación y los datos guardados en GraCT, que es la estructura compacta donde se almacenan los movimientos de los barcos.

4.4.1 Modelo conceptual de datos MongoDB

En este proyecto se utiliza MongoDB, que es un sistema de almacén de datos basado en documentos. Se decidió usar este tipo de base de datos por la flexibilidad que proporciona ya que se pueden almacenar distintos tipos de documentos en la misma colección. Se crearon seis entidades con sus respectivos atributos como se puede ver en la Figura 4.8. A continuación, se explican detalladamente cada uno de los datos de cada entidad que se almacenan en base de datos.

- **Usuario.** Entidad que representa a todo usuario registrado en la aplicación. La información que se almacena sobre este es la siguiente:
  - *login:* nombre identificador del usuario registrado en el sistema. Es único para cada usuario.
  - *contraseña:* código secreto del usuario que le permite acceder a la aplicación. Este código se almacena cifrado en la base de datos.

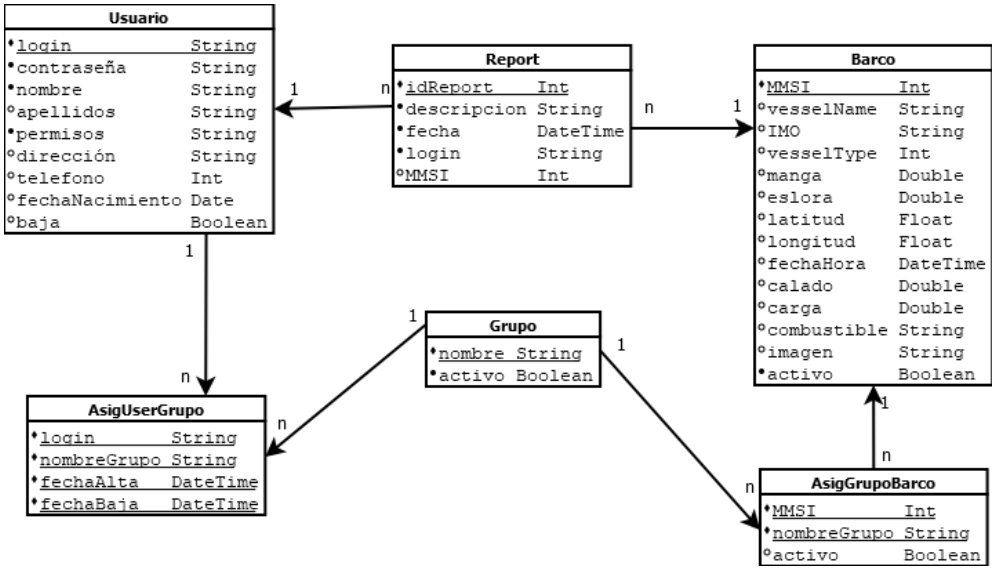


Figura 4.8: Modelo conceptual de datos de MongoDB.

- *nombre:* designación con la que es conocido el nombre públicamente.
- *apellidos:* nombre que le sigue al nombre de pila.
- *permisos:* autorización que permite distinguir entre los usuarios y administradores.
- *dirección:* lugar preciso donde tiene residencia el usuario.
- *teléfono:* número de contacto que permite la comunicación con el usuario.
- *fechaNacimiento:* instante de tiempo en el que nació el usuario.
- *baja:* valor que permite saber si el usuario ha sido dado de baja en el sistema.

- **Barco.** Entidad que representa a todo barco registrado en la aplicación. La información que se almacena sobre él es la siguiente:
  - *MMSI:* iniciales de Maritime Mobile Service Identity Value. Es un número de 8 dígitos que identifica de forma inequívoca a los objetos móviles marítimos.
  - *vesselName:* nombre que se usa para referirse al barco concreto en la emisora de radio.
  - *IMO:* iniciales de International Maritime Organization. Esta formada por las tres letras “IMO” y un número de siete dígitos que es único para cada barco.
  - *manga:* anchura del barco seguiendo las especificaciones de la NAIS (Nationwide Automatic Identification System).
  - *eslora:* longitud del barco medida entra la proa y la popa. Se cuentan las partes no estructurales del barco, siguiendo las especificaciones de la NAIS.

  - *latitud:* distancia angular que hay desde un punto de la superficie terrestre hasta el paralelo del ecuador de la última posición del barco.
  - *longitud:* distancia angular de un punto de la superficie terrestre hasta el meridiano de Greenwich de la última posición del barco.
  - *fechaHora:* instante de tiempo de la última posición registrada del barco.
  - *calado:* altura de la parte sumergida del barco, siguiendo las especificaciones de la NAIS.
  - *carga:* peso que soporta el barco.
  - *combustible:* tipo de combustible que utiliza el barco.
  - *imagen:* representación virtual del barco. Se almacena la url a dicha imagen.
  - *activo:* valor booleano que permite saber si el barco fue dado de baja en el sistema.

- **Report.** Informe/comentario creado por un usuario que permite a los demás usuarios conocer información de interés sobre ese barco. Datos almacenados del informe:
  - *idReport:* número identificativo que crea automáticamente MongoDB cuando se almacena el comentario en base de datos.
  - *descripción:* información del barco introducida por el usuario.
  - *fecha:* instante del tiempo en el que fue creado el comentario.
  - *login:* nombre identificador del usuario registrado en el sistema que creó el comentario.
  - *MMSI:* número identificador del barco del cual se hizo el comentario.

- **Grupo.** Entidad que representa un conjunto de personas y barcos. Para cada uno se almacena la siguiente información:
  - *nombre:* palabra o conjunto de palabras que designan un grupo.
  - *activo:* valor booleano que indica si el grupo fue dado de baja en el sistema.

- **AsigUserGrupo.** Entidad que representa la asignación entre un usuario y un grupo. Los datos que se guardan son los siguientes:
  - *login:* nombre identificador del usuario registrado en el sistema que forma parte del grupo.
  - *nombreGrupo:* nombre identificador del grupo al que fue asignado un usuario.
  - *fechaAlta:* instante de tiempo en el que el usuario fue asignado al grupo.
  - *fechaBaja:* instante de tiempo en el que el usuario fue eliminado del grupo.

- **AsigGrupoBarco.** Entidad que representa la asignación entre un barco y un grupo. Los datos almacenados son los siguientes:
  - *MMSI:* número identificador del barco que fue asignado al grupo.
  - *nombreGrupo:* nombre identificador del grupo al que fue asignado el barco.
  - *activo:* valor booleano para saber si el barco sigue formando parte del grupo o no.

4.4.2 Datos almacenados en GraCT

En esta sección se detallarán los datos que se almacenan en la estructura GraCT.

- **Identificador del objeto:** se almacena una lista con todos los identificadores de los objetos. Este identificador coincide con el MMSI que se almacena en la base de datos MongoDB.
- **Punto:** cada objeto envía los puntos por los que pasa en cada instante de tiempo. Cada punto se corresponde a celdas de una representación ráster en el espacio. Un ráster es una estructura, normalmente cuadrada, dividida en celdas. Cada celda posee un valor, en este caso entero para representar valores discretos. Los rásteres se almacenan en forma de lista ordenada de los valores de celda. [23]
- **Instante de tiempo:** los objetos envían su última localización y el instante de tiempo en el que pasaron por ese punto. Estos instantes de tiempo tienen una precisión de un minuto.

4.5 API REST

En esta sección se muestran los endpoints que se crearon para cada una de las entidades del servicio, *usuario, barco, grupo y report*, que se pueden ver en el Cuadro 4.3, Cuadro 4.2, Cuadro 4.4 y Cuadro 4.5 respectivamente, junto con los métodos que resuelven dichos endpoints. Los endpoints son las URLs del servicio de un API que responden una petición.

Los endpoints de nuestra API se encuentran en el directorio *routes* del servicio, cuya estructura se explica en la Apartado 5.2.2.

Petición	**Endpoint**	**Método**
POST	/authenticate	autenticarse
POST	/usuarios/registrar	registrar
GET	/account	getAccount
GET	/usuarios	findAll
GET	/usuarios/baja	findUsuariosBaja
GET	/usuarios/login	findByLogin
PUT	/usuarios/editar/login	editarPerfil
PUT	/usuarios/baja/login	baja
PUT	/usuarios/alta/login	alta
PUT	/usuarios/cambiar-pass/login	cambiarContraseña

Cuadro 4.2: Endpoints de la entidad *usuario*.

Petición	**Endpoint**	**Método**
POST	/barcos/nuevo	create
GET	/barcos	findAll
GET	/barcos/MMSI	findByMMSI
GET	/barcos/one/vesselName	findByVesselName
PUT	/barcos/editar/MMSI	editarBarco
PUT	/barcos/baja/MMSI	bajaBarco
GET	/ruta/lastLocations	getLastLocations
GET	/ruta/trajectory/MMSI/fechaInicio/fechaFin	getTrajectory
POST	/point/t/knum/latitud/longitud	consultarPunto
POST	/region/t/latitudMax/longitudMin/latitudMin/longitudMax	regionTimeSlice
POST	/region/timeInit/timeEnd/latitudMax/longitudMin/latitudMin/longitudMax	regionTimeInterval
POST	/barcos/search/MMSI/ vesselType/vesselName/eslora/manga/calado/carga	busquedaAvanzada

Cuadro 4.3: Endpoints de la entidad *barco*.

Petición	Endpoint	Método
GET	/grupos	findAll
GET	/grupos/nombre	findByNombre
POST	/grupos/nuevo	create
PUT	/grupos/baja/nombre	baja
POST	/grupos/asignar-usuario	asignar
POST	/grupos/quitar-usuario/nombre/login	quitarUsuario
GET	/asignaciones-usuario-grupo/nombreGrupo	findUsuariosAltaBy-Grupo
GET	/asignaciones-usuario-grupo/login/grupos	findGruposByUsuario
POST	/grupos/asginar-barco	asignarBarco
GET	/asignaciones-barco-grupo/nombreGrupo	findBarcosByGrupo
GET	/asignaciones-barco-grupo/MMSI/grupos	findGruposByBarco
POST	/grupos/quitar-barco/nombreGrupo/MMSI	quitarBarco

Cuadro 4.4: Endpoints de la entidad *grupo*.

Petición	Endpoint	Método
GET	/reports	findAll
POST	/reports/nuevo	create
GET	/reports/login	findByUser

Cuadro 4.5: Endpoints de la entidad *report*.

Capítulo 5

Diseño

5.1 Arquitectura tecnológica del sistema

Para cada uno de los componentes de la Figura 4.1, se describen en esta sección las tecnologías utilizadas en cada uno de ellos. En la Figura 5.1 se puede ver el diagrama de componentes de esta arquitectura tecnológica.

- Para la implementación del cliente se utilizó el framework Vue.js y la librería Leaflet de JavaScript para la creación de mapas. Esta capa se comunica con el servicio a través de peticiones HTTP para las cuales se utilizó la tecnología Axios.

- En la implementación del servicio se utilizó Express.js, que es un framework de Node.js que nos permite crear el servidor con facilidad, ya que proporciona funcionalidades como el enrutamiento, para recibir peticiones del cliente y hacer llamadas a las operaciones de los controladores de las entidades correspondientes para acceder a los datos de la base de datos.

- Como base de datos se utiliza MongoDB. Esta es una base de datos NoSQL y los datos se almacenan como documentos. Además, se utilizó el módulo *mongoose* para definir los esquemas de las entidades que se asignan a un documento MongoDB y este módulo, a su vez, realiza la conexión con la base de datos, permitiendo al servidor el acceso a los datos almacenados. Este módulo proporciona métodos y funcionalidades para trabajar con MongoDB.

- Finalmente para realizar la integración con la estructura compacta GraCT, como esta estructura está diseñada en el lenguaje C++, fue necesaria la creación de una librería propia que está basada en C++ y en N-API, que es un módulo de Node.js que permite la comunicación del servicio con GraCT.

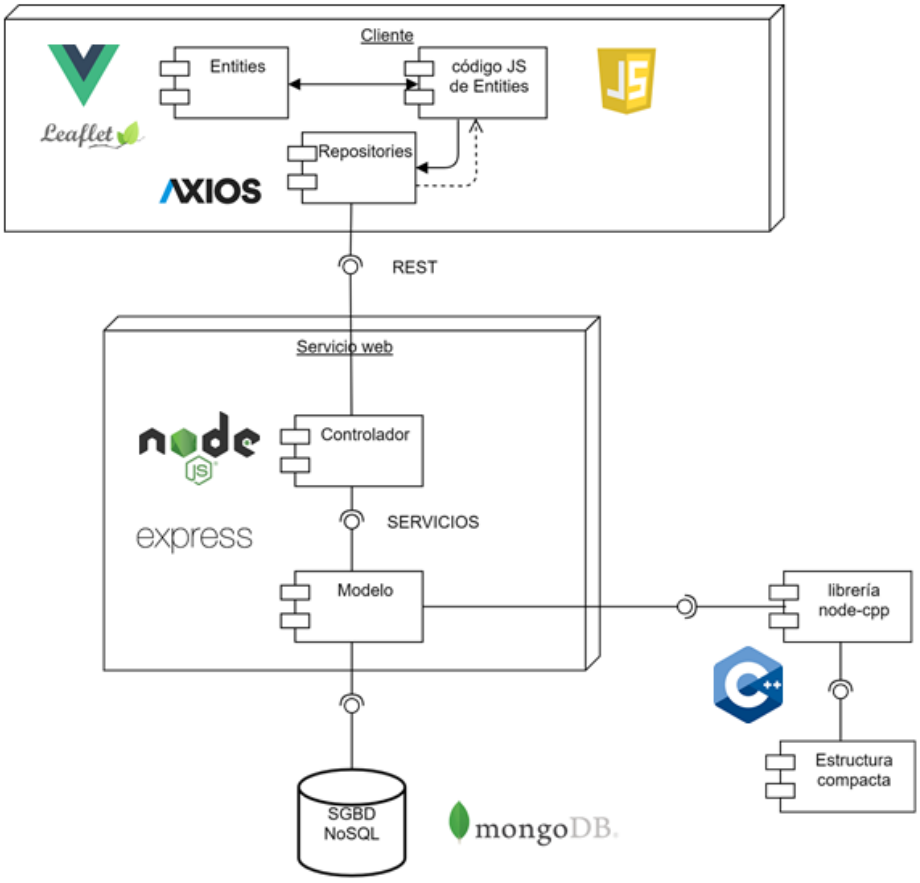


Figura 5.1: Diagrama de componentes de la arquitectura tecnológica del sistema.

5.2 Diseño de la aplicación

Esta sección se divide en tres secciones distintas donde se explican detalladamente los diferentes componentes del sistema que se muestran en la figura Figura 4.1. Se describe la trazabilidad entre el Capítulo 4 y la implementación de la aplicación.

5.2.1 Cliente web

En esta sección se detallan los componentes del cliente web, describiendo los paquetes que contiene y la conexión existente entre cada uno de ellos.

Estructura del cliente. La estructura inicial del proyecto contiene un fichero *package.json* que incluye las librerías y las herramientas que se utilizan en el proyecto. Además, está estructurada en los siguientes directorios:

- *src/assets:* es el paquete donde se encuentran los ficheros públicos de la aplicación. En este caso solo hay imágenes.
- *src/common:* paquete que contiene las utilidades comunes de la aplicación. En este caso, utilizado por la entidad "Login", cuando un usuario se autentica se genera el token correspondiente, este se valida y, en caso de que sea válido, es almacenado en la base de datos.
- *src/plugins:* es el paquete donde se definen las funcionalidades adicionales que utiliza la interfaz.
- *src/components:* son los componentes comunes que no están relacionados con las entidades. De manera que se puedan reutilizar sin necesidad de repetir código.
- *src/entities:* son los componentes que están relacionados con las entidades de la aplicación. Utilizan los ficheros del paquete *Repositories* para hacer la llamada a las funciones.
- *src/repositories:* son los clientes REST de las entidades. Se encargan de hacer las peticiones HTTP al servicio, a través de ellos se mantiene la conexión entre la capa cliente y la capa de servicio.

En la Figura 5.2 se puede ver un diagrama de paquetes de la estructura del cliente detallada anteriormente.

5.2.2 Servicio web

En esta sección se detallan los componentes del servicio web, describiendo brevemente los paquetes que contiene y la conexión que existe entre ellos.

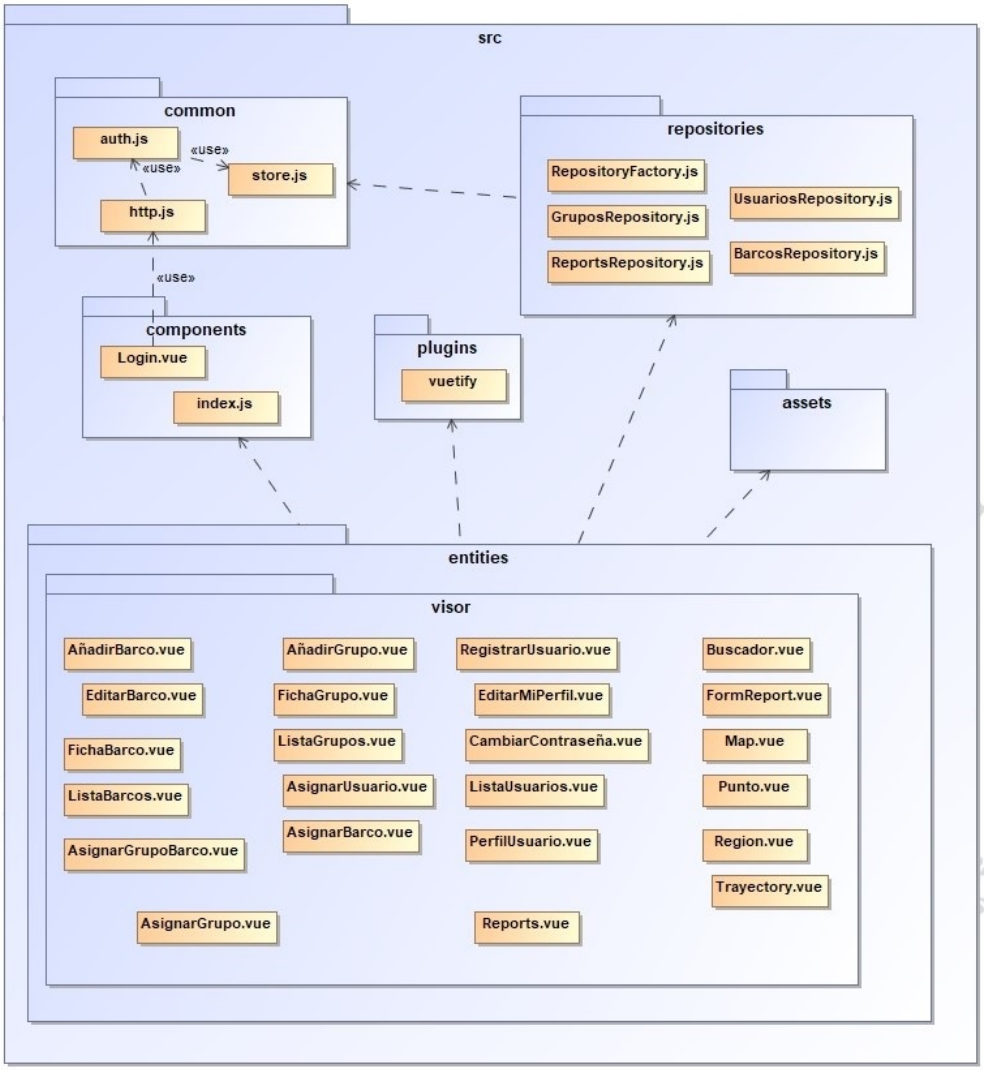


Figura 5.2: Diagrama de paquetes del cliente web.

Estructura del servicio. La estructura inicial del proyecto está compuesta por un fichero *package.json* que contiene los paquetes que se instalan en el proyecto y sus dependencias, un archivo *database.js* en donde se crea la conexión a la base de datos y un archivo *app.js* en donde se crea el servidor web con Node.js y la configuración con Express.js, así como la carga del módulo de Node.js, "greet.node", para poder hacer consultas contra estructura GraCT. Además, se estructura en los siguientes directorios:

- *middleware:* este directorio contiene un fichero para validar la autenticación del usuario.
- *multer:* incluye un fichero para gestionar el almacenamiento de las imágenes de los barcos, es decir, el directorio en el que se almacenan, la creación de la url que se almacenará en la base de datos y los tipos de fichero permitidos.
- *models:* existe un archivo por cada una de las entidades, en donde se crea el esquema, que es la especificación de los atributos del documento de la colección que se va a crear, y el modelo de la entidad. Para la realización de las operaciones a la base de datos se hace uso de esta abstracción.
- *controllers:* contiene un archivo controlador para cada entidad. Son los encargados de comunicarse con los servicios de la capa modelo. Cada clase implementa un endpoint REST.
- *services:* existe un archivo por cada entidad y en él se implementan las operaciones que se realizan contra la base de datos. Para hacer estas peticiones se utiliza el modelo de la entidad correspondiente.
- *routes:* incluye los archivos donde se definen las rutas a las peticiones que responderá la aplicación.

En la Figura 5.3 se muestra un diagrama de paquetes de los componentes del servicio web explicados anteriormente.

5.2.3 Diseño API Node-CPP

Como la estructura compacta GraCT está diseñada en C++, fue necesaria la creación de una librería, utilizando las tecnologías C++ y N-API, que permitiera la integración del back-end con GraCT. A la estructura GraCT se accede a través de esta librería en donde están implementadas las siguientes operaciones:

- **Obtener las últimas posiciones de los barcos**. Esta consulta se encarga de obtener todos los barcos que están almacenados en GraCT junto con su última posición y el instante de tiempo en el que fue emitido el punto. Este dato es almacenado en la base de datos conceptual de la aplicación para conseguir accesos rápidos.

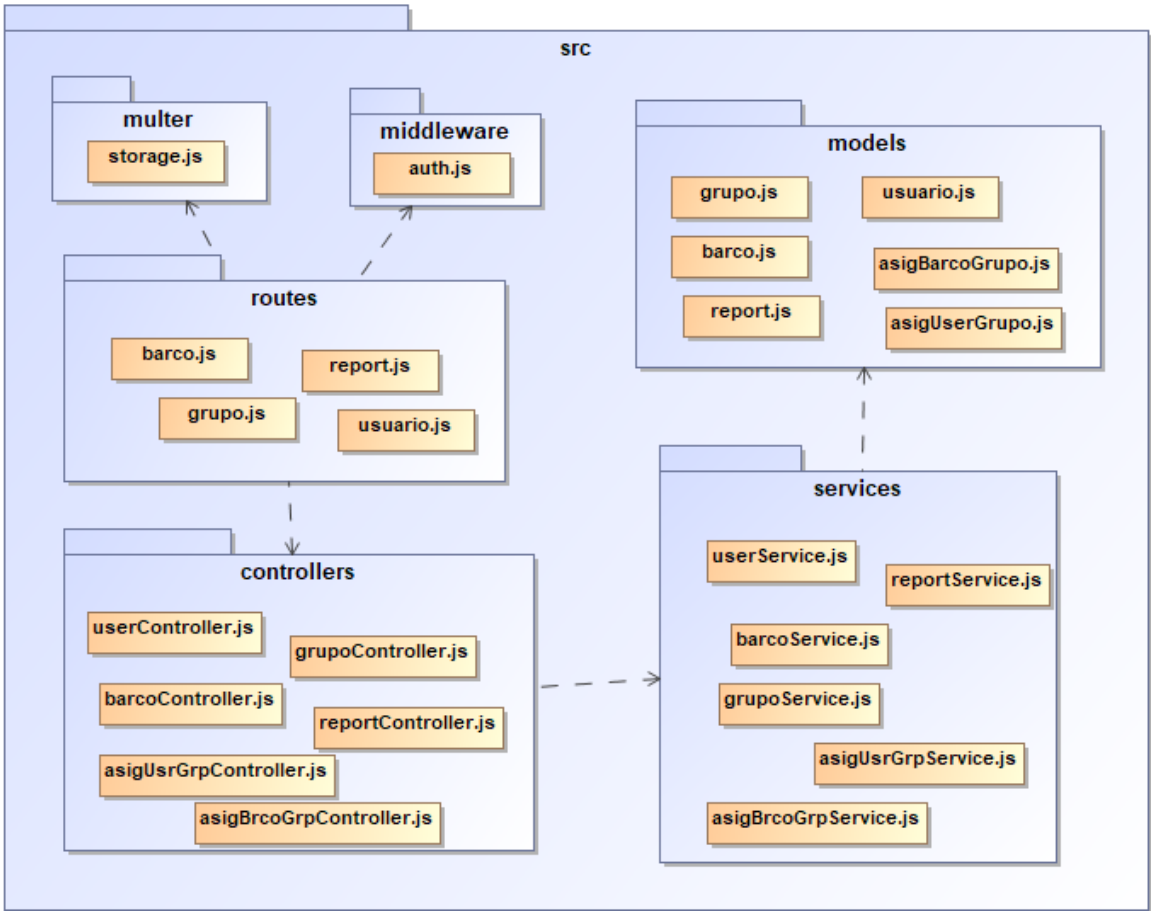


Figura 5.3: Diagrama de paquetes del servicio web.

- **Obtener la trayectoria de un barco**. Esta consulta obtiene los puntos recorridos por un barco entre los instantes de inicio y fin dados por el usuario, una vez que se compruebe que son válidos.
- **Obtener los barcos más próximos a un punto**. Consulta que permite conseguir un número determinado de barcos indicados por el usuario que están más próximos a una coordenadas dadas. Se comprueban que tanto las coordenadas como el instante de tiempo sean válidos.
- **Obtener los barcos de una región en un instante de tiempo**. En esta consulta se comprueba que las coordenadas del área seleccionada por el usuario y el instante de tiempo sean válidos. Posteriormente, se obtienen los barcos de la región junto con las coordenadas en las que estuvo en dicho instante de tiempo.
- **Obtener la trayectoria de los barcos de una región**. Se comprueba que tanto las coordenadas como el intervalo de tiempo indicado por el usuario sean válidos. Para cada barco que pueda ver el usuario se obtiene la trayectoria que este realizó entre los instantes de inicio y fin indicados. No se pueden coger solo aquellas posiciones de la ruta del barco que estén dentro de la región ya que podría aparecer dividida, porque durante el período de tiempo el barco pudo haber salido de la región y haber vuelto a entrar; esto daría lugar a confusión.

Además de estas consultas, existen los métodos *buildGraCT* y *loadGraCT* que se utilizan para construir y cargar el índice de GraCT respectivamente, el cual permite acceder a esta estructura. El método *buildGraCT*, para crear el índice, tarda unos minutos, por lo que esta operación no se puede ejecutar cuando se lanza el servidor, ya que el timeout de este último es menor que el tiempo de ejecución del método que construye el índice. Por lo tanto, se creó un script para construirlo y ejecutarlo manualmente. También se creó un script para obtener las últimas posiciones de todos los barcos y actualizar la base de datos conceptual de la aplicación con esta información.

Cabe destacar que la librería no solo hace las llamadas a las funciones de GraCT, sino que se han hecho en ella algunas optimizaciones en cuanto a la obtención de solo los datos de aquellos barcos que pueda ver el usuario, y no de todos los que cumplan las condiciones, y en cuanto a la consulta de la obtención de las trayectorias de los barcos de una región en un intervalo de tiempo. Esto se explicará en detalle en el Capítulo 6.

5.2.4 Funcionamiento de los componentes al realizar una petición

Una vez definidos los tres componentes claves de la aplicación, se explica brevemente como sería su funcionamiento cuando un usuario realiza una petición. Una vez que el usuario

está autenticado en la aplicación y realiza una petición, esta hace una llamada a la función ubicada en el paquete *Repositories* de la entidad correspondiente. Aquí, se hace una petición HTTP al servicio utilizando para ello la tecnología *Axios*. En el servicio se definen las rutas en donde se encuentra la lógica a ejecutar, accediendo al controlador que es el encargado de llamar a la función correspondiente del paquete *services* en donde están implementadas las operaciones sobre la base de datos, utilizando el modelo de la entidad para acceder a ella en caso de que los datos que se deseen obtener estén almacenados en la base de datos de mi aplicación. En otro caso, se realizaría la consulta a la estructura GraCT. Para poder hacer estas consultas, es necesario importar en el servicio el módulo "greet" de node, que puede ser almacenado en una variable. Primero, se necesita crear y cargar el índice de GraCT en el servicio y después se pueden hacer las llamadas a las funciones a través de la librería creada. Cabe destacar que el cliente puede ejecutar un script para crear el índice y este se carga cuando se lanza el servicio.

Capítulo 6

Implementación y pruebas

6.1 Implementación

En esta sección se explican los datos que se han utilizado y la importación de estos en la base de datos conceptual de la aplicación, así como algún algoritmo complejo que se ha implementado y las optimizaciones que se han hecho y que no se han detallado en capítulos anteriores.

6.1.1 Importación de datos

Es importante destacar que en esta aplicación se ha utilizado un data set con la información real de los barcos [24]. Disponemos de un archivo *zip* que incluye varios ficheros, uno de ellos es el data set original y después hay versiones recortadas desde 100MB, 500MB y 1GB. Estos ficheros son los que contienen la información geográfica de los barcos y la cuál será almacenada en GraCT. Además, incluye un archivo *csv* con los datos de cada barco. Este fichero *csv* se importa en la base de datos conceptual de la aplicación, en nuestro caso, MongoDB. Al utilizar un data set real, habrá barcos que carezcan de información, existiendo, simplemente, los datos más relevantes, entre ellos el número identificador del barco. Para cada uno de ellos, se almacenará en MongoDB su última posición conocida, que será extraída de GraCT, ya que queremos tener accesos rápidos. Se espera que se haga este volcado de datos una vez cada noche cuando se actualicen los datos de GraCT, para lo cual se ha definido una operación de node que se puede ejecutar de manera externa con un script de node.

Una vez hecho este volcado de datos, la petición para obtener la última posición conocida de los barcos se realiza contra la base de datos conceptual. Para poder realizar consultas desde el servicio a la estructura compacta, es necesario que, primero, se cree el índice de GraCT en el servicio. Como es una gran cantidad de datos la que se almacena en esta estructura, se decidió que la creación de este índice se realice ejecutando un script externo al servicio lanzado y una vez se lanza el servicio, se carga el índice en él. Para poder hacer esta operación, como bien

se comentó en el Capítulo 5 en la Apartado 5.2.3, se necesita importar el módulo "greet" de Node.js que se tiene con la librería "gract-module" que se creó para conectarse a la estructura compacta.

6.1.2 Configuración de la librería node-cpp

En esta sección se explica cómo se configuró la librería para que permitiera la conexión entre nuestro proyecto y la estructura compacta.

- Primero se **crea el proyecto "gract-module"** utilizando el módulo *node-addon-api* (N-API) de Node.js. Este es un fichero binario que es compilado utilizando el lenguaje C++. Para utilizar las características de Native Addon es necesario instalar el paquete *npm* y *node-gyp* de npm. Se ejecuta el comando *npm init -y* para inicializar el paquete "package.json" y creamos el módulo Native Module con el nombre de "greet". Esta propiedad tiene que estar en el package.json y a continuación, se pueden instalar todas las dependencias necesarias, en nuestro caso, *node-addon-api* (N-API). El hecho de utilizar N-API nos facilita no tener que instalar dependencias, es suficente con importarlas en la cabecera del fichero [25]. Para crear el módulo "greet" se ejecuta el comando *node-gyp configure*, que genera el modelo de código dentro del directorio *build*.

- Una vez que se construyeron las instrucciones para el compilador de C++, es cuando se procede a la **creación del módulo "greet"** con el comando *node-gyp build*, el cual crea el fichero "greet.node" y lo sitúa en el directorio *build/Release*, que posteriormente será importado en nuestro servicio para poder utilizar las operaciones exportadas.

- Finalmente, **se crean y exportan las consultas a GraCT**. En el fichero "index.cpp" donde se implementan todas las funciones para consultar los datos de GraCT, se utiliza la función *NODE_API_MODULE(greet, Init)*, la cual hace que el módulo "greet" creado anteriormente, se inicialice con las operaciones que exporta la función "Init", que serían todas las consultas que se pueden hacer contra GraCT, mencionadas en la Apartado 5.2.3, de manera que cuando este módulo se importe en nuestro código JavaScript, se podrá acceder a las funciones exportadas de la librería. La función Init devuelve las operaciones en un objeto, indicando por qué función fue implementada cada funcionalidad.

Para poder consultar los datos de GraCT, se crea y almacena el índice en una variable global, que una vez inicializado, será utilizado en la implementación de las funciones.

6.1.3 Implementación de las optimizaciones en las consultas a GraCT

Existen dos problemas destacables en el proyecto, los cuales se han resuelto de la forma más óptima posible y son los que se explican a continuación.

Por una parte, nuestra aplicación asigna a los usuarios y barcos a diferentes grupos, de manera que los usuarios solo tendrán acceso a la información de aquellos barcos que pertenezcan a sus grupos. Como **GraCT** solo almacena información geográfica de los barcos, **no puede gestionar ni los usuarios, ni los barcos que son visibles para ellos**.

Por otra parte, **el visor no es capaz de dibujar todas las trayectorias de los barcos que pasaron por una región con un buen tiempo de respuesta**, ya que cada una de ellas contiene un gran cantidad de puntos.

Como consecuencia de estos dos problemas, la implementación de la librería que comunica la estructura compacta con nuestro back-end, no solo se encarga de hacer las llamadas a GraCT sino que trata de solucionar esta problemática. Un ejemplo en donde se resuelven ambos es en la consulta de la trayectoria de los barcos que pasaron por una región entre dos instantes de tiempo. Estas dos optimizaciones se explican a continuación dividas en dos partes.

- **Filtrado de barcos del usuario**. Para poder obtener la trayectoria de los barcos que son visibles para el usuario, es necesario que el servicio le envíe a la librería node-cpp un array con estos. Una vez que se comprueba que las condiciones son válidas, los instantes de tiempo y la región, se convierte este array a un 'unordered_map' para facilitar el filtrado de barcos (un 'unordered_map' funciona como un HashMap en Java). A continuación, se obtienen los identificadores que cumplan las condiciones que son los que se utilizan para consultar en GraCT. Posteriormente, para cada uno de ellos se obtiene el identificador original que sería el equivalente al MMSI que se almacena en la base de datos conceptual, y se comprueba si este identificador existe en la lista de los barcos del usuario. A continuación, se muestra el código fuente en donde se ejecutan estas operaciones.

```
1     if(ok_time && ok_region){
2         array_to_map(barcosUser, map);
3         //Obtención de los identificadores de los barcos que están
    en el intervalo de tiempo indicado.
4         auto ids = m_gract_global.time_interval(t_s, t_e, r_q);
5         uint j = 0;
6         for(uint64_t i = 0; i < ids.size(); ++i){
7             //Obtención del identificador MMSI del barco
8             auto oid = m_gract_global.get_original_id(ids[i]);
9             auto exist = exists(map, oid);
10            Napi::Array arrayCoords = Napi::Array::New(env);
11
```

- **Obtención de la trayectoria de los barcos.** Continuando con lo explicado anteriormente, si el barco existe en la lista de los barcos del usuario, se obtiene la trayectoria realizada en el intervalo de tiempo indicado. Como los puntos que emite un barco son a cada minuto, es muy posible que la cantidad de puntos de la trayectoria almacenados en GraCT sea muy elevada, entonces solo se conseguirán algunos puntos del recorrido. Para ello, una vez obtenida la ruta, como se puede conseguir su tamaño, esta se divide entre la cantidad de puntos que se quieren adquirir de la trayectoria y el resultado obtenido es utilizado como salto en el bucle *for*. De este modo, solo se consiguen parte de los puntos y se le envía al servicio una menor cantidad de datos, ganando así en tiempo de respuesta. A continuación, se muestra el código fuente que resuelve este problema.

```
1    if(exist){
2        Napi::Object obj  = Napi::Object::New(env);
3        std::vector<data_traj> traj;
4        //Se obtiene la trayectoria del barco.
5        m_gract_global.search_trajectory(ids[i], t_s, t_e, traj);
6        uint sizeT = traj.size();
7        obj["oid"] = oid;
8        found[j] = obj;
9        //Se comprueba que el tamaño de la trayectoria sea mayor
    que el número de puntos que se quieren obtener.
10           if(sizeT > ptos){
11               uint salto = sizeT / ptos;
12               uint x = 0;
13               for(uint64_t z = 0; z < traj.size(); z+=salto){
14                   Napi::Object objCoords = Napi::Object::New(env);
15                   //Se obtienen las coordenadas de cada punto de
    la trayectoria
16                   auto p =
    m_gract_global.get_coords_from_point(point(traj[z].m_x,
    traj[z].m_y));
17                   objCoords["lat"] = p.m_y;
18                   objCoords["lng"] = p.m_x;
19                   arrayCoords[x] = objCoords;
20                   ++x;
21               }
22               //Se crea el objeto con el array de coordenadas de
    la trayectoria
23               obj["coords"] = arrayCoords;
24               found[j] = obj;
25               ++j;
26
```

En la Figura 6.1 se puede ver la secuencia de las operaciones que se utilizaron para resolver

los problemas que se explicaron anteriormente de la función creada para obtener la trayectoria de los barcos que pasaron por una región en un intervalo de tiempo.

6.2 Pruebas

Tanto para la capa cliente como para el servicio, se han probado todos los casos posibles de las operaciones con la herramienta Postman, para verificar que el funcionamiento de estas era el esperado. En el Cuadro 6.1, se muestran algunas de las pruebas realizadas con el método HTTP utilizado, el endpoint y el cuerpo de la petición en los casos en los que fuera necesario. Cabe destacar que en esta tabla no se muestran todos los casos posibles que se han probado.

En la librería en donde se hacen las consultas a la estructura compacta, primero se han utilizado valores de prueba para comprobar que funcionaban correctamente, creando un fichero en donde se ejecutaran las funciones. Posteriormente, se implementaron en el servicio las operaciones que llamaran a estas funciones.

Método HTTP	**Endpoint**	**Cuerpo de la petición**
POST	https://localhost:8080/visor/barcos/nuevo	MMSI, vesselName, IMO, vesselType, eslora, manga, calado, carga, combustible, image
PUT	https://localhost:8080/visor/barcos/editar/MMSI	vesselName, eslora, manga
PUT	https://localhost:8080/visor/barcos/baja/MMSI	-
GET	https://localhost:8080/visor/barcos/MMSI	-
GET	https://localhost:8080/visor/barcos	-
GET	https://localhost:8080/visor/account	token
POST	https://localhost:8080/visor/authenticate	login, contraseña
POST	https://localhost:8080/visor/registrar	nombre, apellidos, login, contraseña, fechaNacimiento, provincia, telefono
GET	https://localhost:8080/visor/usuarios/login	-
GET	https://localhost:8080/visor/usuarios	-
PUT	https://localhost:8080/visor/usuarios/baja/login	-
POST	https://localhost:8080/visor/grupos/nuevo	nombre
GET	https://localhost:8080/visor/grupos	-
PUT	https://localhost:8080/visor/grupos/baja/nombreGrupo	-
GET	https//localhost:8080/visor/grupos/nombre	-

POST	https://localhost:8080/visor/grupos/asignar-usuario	login, nombreGrupo
GET	https://localhost:8080/visor/asignaciones-usuario-grupo/nombreGrupo	-
POST	https://localhost:8080/visor/grupos/quitar-usuario/nombreGrupo/login	-
GET	https://localhost:8080/visor/asignaciones-usuario-grupo/login/grupos	-
POST	https://localhost:8080/visor/grupos/asignar-barco	MMSI, nombreGrupo
POST	https://localhost:8080/visor/grupos/quitar-barco/nombreGrupo/MMSI	-
GET	https://localhost:8080/visor/asignaciones-barco-grupo/nombreGrupo	-
GET	https://localhost:8080/visor/asignaciones-barco-grupo/MMSI/grupos	-
POST	https://localhost:8080/visor/reports/nuevo	descripcion, login, MMSI
GET	https://localhost:8080/visor/reports	-
GET	https://localhost:8080/visor/reports/login	-

Cuadro 6.1: Tabla de las pruebas realizadas con Postman.

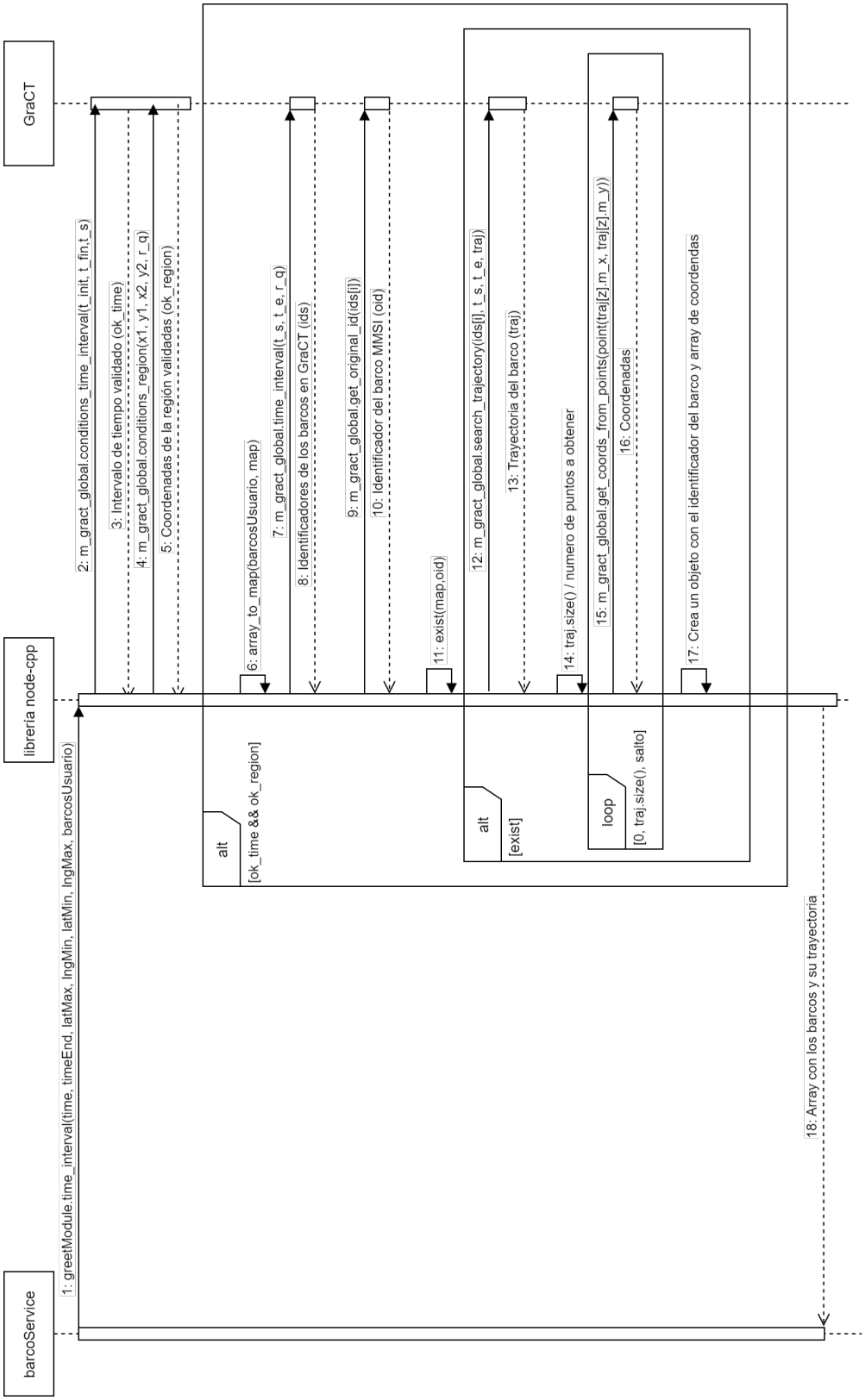


Figura 6.1: Diagrama de secuencia para obtener la trayectoria de los barcos de una región.

Capítulo 7

Solución desarrollada

En esta sección se explica cómo funciona la aplicación desarrollada, haciendo una visita guiada por las principales funcionalidades acompañada de capturas de pantalla de la aplicación final. Se divide en tres secciones, en una se detallan las consultas que puede hacer un usuario sobre los barcos, en otra el acceso al perfil del usuario y en la última, se explican las funcionalidades de gestión de usuarios, barcos y grupos de las cuales se encarga el usuario.

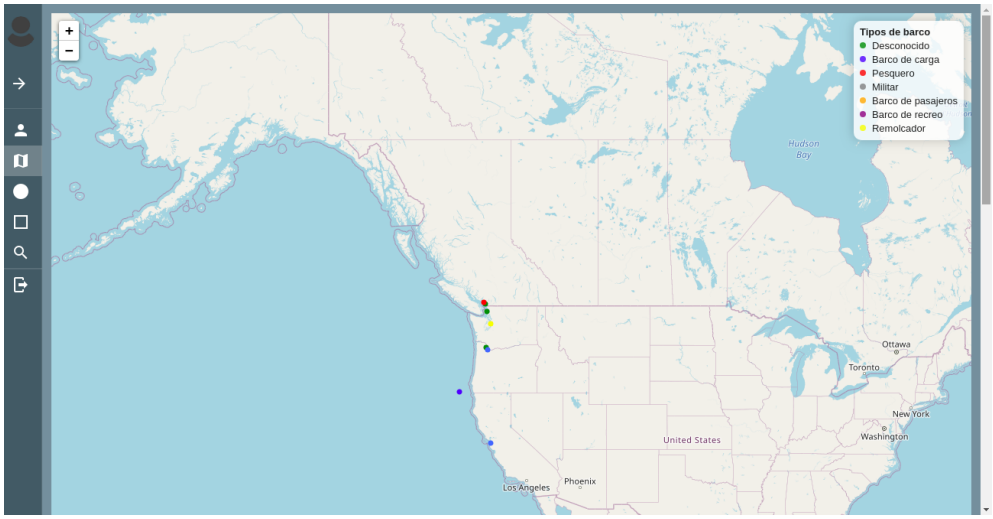
7.1 Consultas sobre barcos permitidas a los usuarios

Esta sección se divide en cuatro subsecciones en donde se explican como funcionan las diferentes consultas que puede hacer el usuario sobre los barcos.

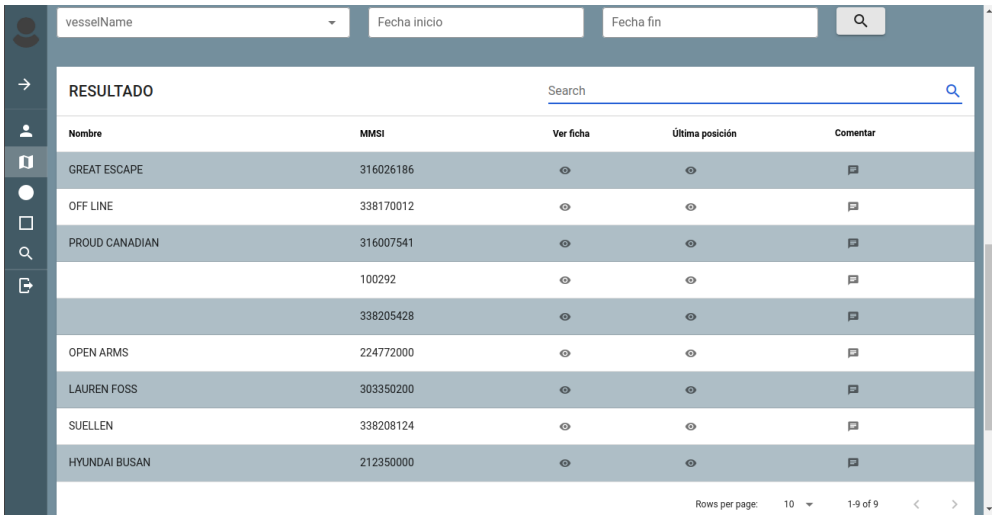
7.1.1 Búsqueda de la trayectoria de un barco

Cuando el usuario se autentica en la aplicación, la pantalla principal es la que se muestra en la Figura 7.1, en donde se visualiza un visor de mapas que muestra los barcos que son visibles para un usuario, es decir, aquellos que pertenecen a los mismos grupos a los que pertenece el usuario. Como existen diferentes tipos de barcos, se le ha asignado a cada tipo un color diferente, los cuales están definidos en la leyenda que se puede ver en el visor de mapas en la Figura 7.1a. En esta pantalla principal, se muestra también una tabla con los mismos barcos que se están visualizando en el mapa, y desde ella, el usuario puede acceder a funcionalidades como ver la ficha de un barco concreto, la cual contiene la información del barco y un enlace para ver su última posición conocida y otro para ver la trayectoria realizada en las últimas 5 horas, como se puede ver en la Figura 7.2. También puede acceder directamente desde la tabla a la última posición del barco, que se visualizaría en mapa, así como crear un comentario sobre un barco concreto. Además, el usuario también puede ver sus propios comentarios, así como los comentarios de los barcos, de sus grupos, que han hecho otros usuarios. Cabe destacar que como la cantidad de barcos puede ser elevada, en la tabla existe un buscador que le facilita al

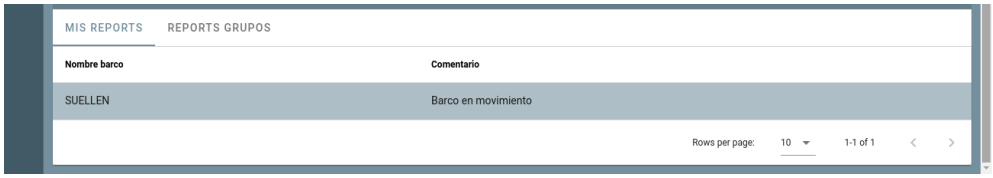
usuario buscar un barco concreto, sin necesidad de tener que indagar por toda la tabla. Este buscador le permite al usuario filtrar los barcos tanto por MMSI como por nombre.



(a) Visor de mapas



(b) Tabla resultado con los barcos que se muestran en el visor de mapas.



(c) Tabla con los reports de los barcos.

Figura 7.1: Pantalla principal de la aplicación.

En esta pantalla principal, el usuario también puede consultar la trayectoria de un barco concreto. Como se puede ver en la Figura 7.1b, existen tres inputs que le permiten al usuario

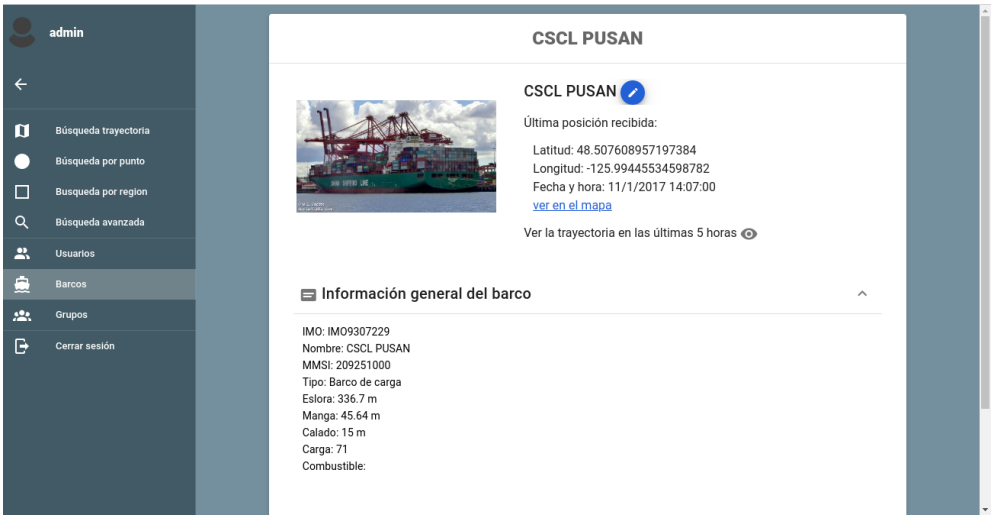


Figura 7.2: Ficha de un barco concreto.

seleccionar el nombre del barco y los instantes de tiempo de inicio y fin entre los cuales quiere obtener la ruta realizada por el barco. Una vez que el usuario hace la petición, se visualiza en el mapa dicha ruta, como se puede ver en la Figura 7.3 , en caso de que exista. En caso de que no exista ningún punto por el que haya pasado ese barco entre los instantes de tiempo, se mostrará un mensaje informando al usuario de que el barco no ha seguido ninguna ruta en dicho intervalo.

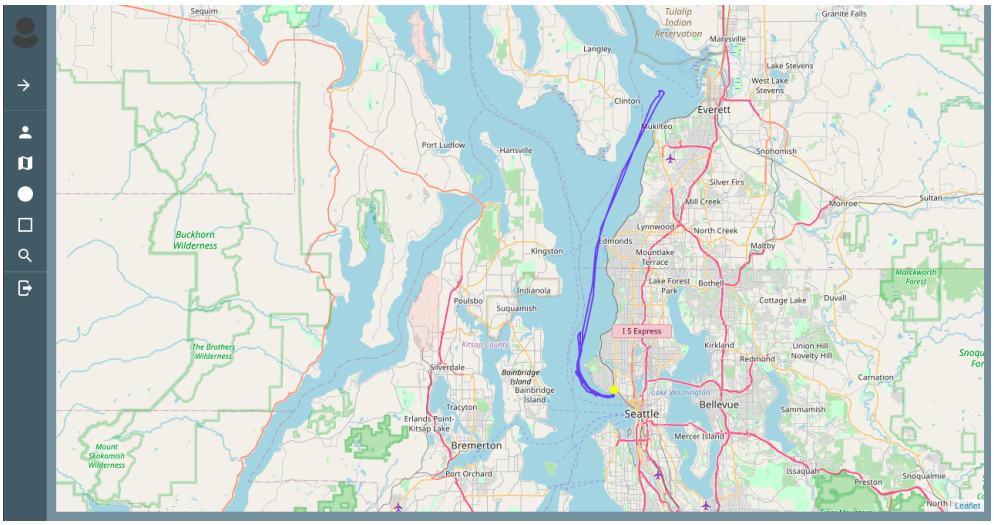


Figura 7.3: Resultado de la búsqueda de la trayectoria de un barco.

7.1.2 Búsqueda por punto

Otra de las consultas que el usuario puede realizar es la obtención de un número de barcos más próximos a un punto del mapa indicado por el usuario. En la Figura 7.4 se puede ver la pantalla que permite hacer esta consulta. En ella se muestra un visor de mapas y cuatro inputs donde el usuario indica el instante de tiempo en el cual los barcos estuvieron próximos al punto que seleccione en el mapa y el número de ellos que quiere visualizar en el visor de mapas. Las coordenadas del punto seleccionado se muestran en los inputs editables "latitud" y "longitud". Cabe destacar que en la esquina inferior izquierda del mapa existe un contenedor que va mostrando las coordenadas por donde se está desplazando el ratón en el mapa. En la Figura 7.5 se puede ver el resultado obtenido de esta consulta. En la Figura 7.5a se visualizan los barcos en el mapa y además se muestra una tabla con los barcos resultado de la consulta, que se puede ver en la Figura 7.5b, en donde el usuario puede acceder a la ficha del barco, así como ver la última posición de un barco concreto. Para ver la última posición se mostrará también un minimapa de contexto que le facilite al usuario conocer la localización en donde se encuentra el barco, y además, si selecciona el barco se le abrirá un 'popup' con el nombre, MMSI y la fecha y hora de la última posición de barco, así como también un enlace a la ficha de este para proporcionarle al usuario un acceso rápido. Esto se puede ver en la Figura 7.6. Este 'popup' se abrirá siempre que el usuario seleccione un barco desde cualquiera de los mapas.

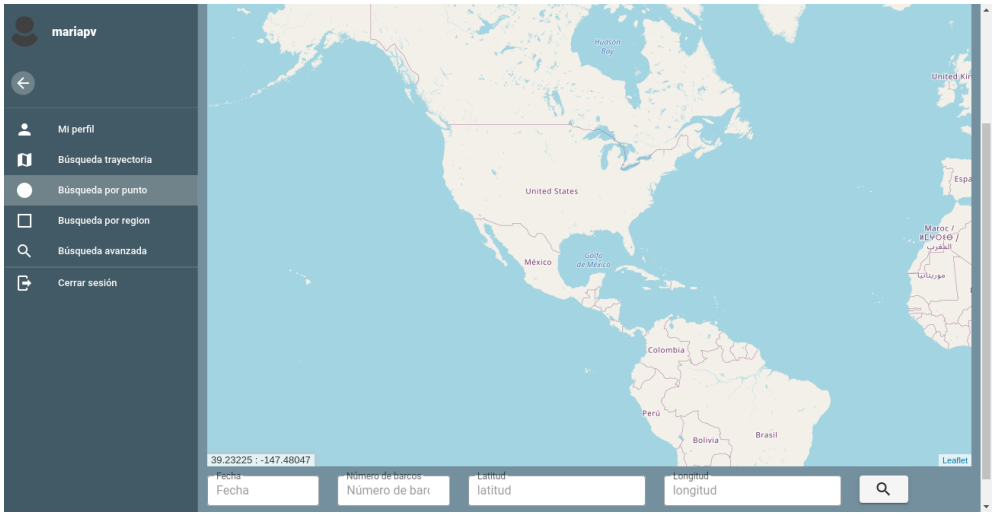
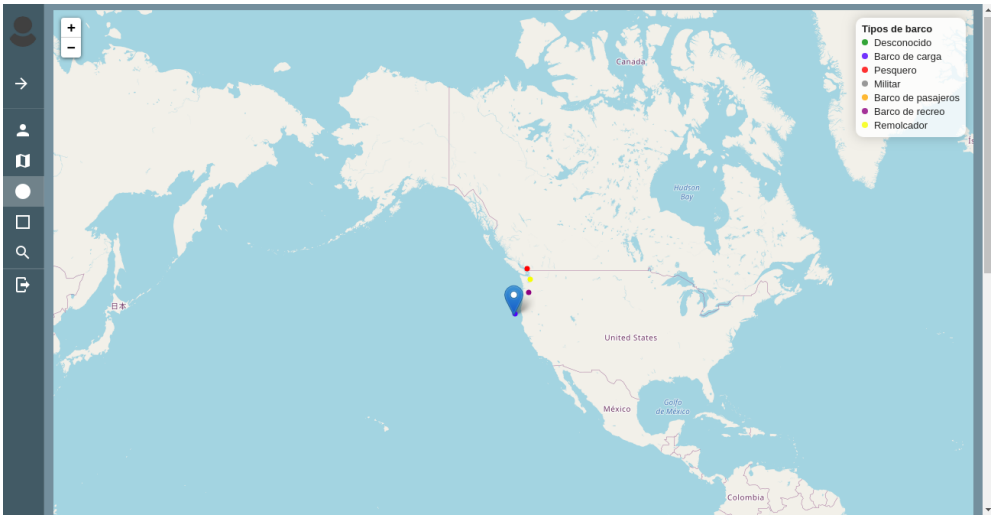


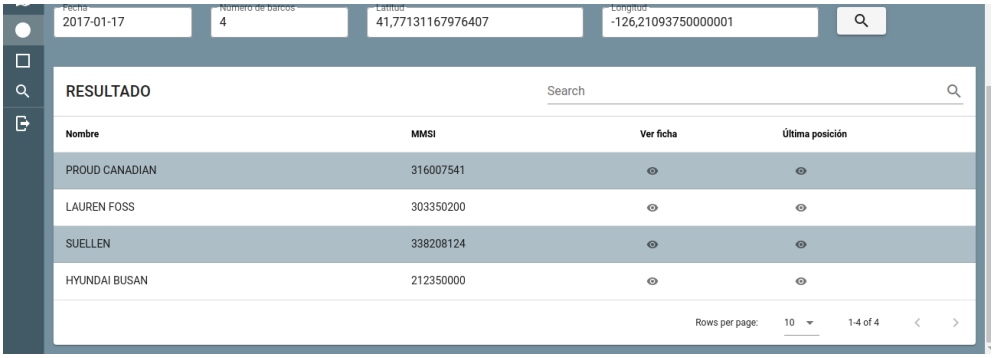
Figura 7.4: Consultar los barcos más próximos a un punto en el mapa.

7.1.3 Búsqueda por región

En la Figura 7.7 se muestra la pantalla para consultar los barcos de una región. Aquí se pueden realizar dos consultas diferentes.



(a) Visor de mapas con los barcos resultado de la consulta.



(b) Tabla con los barcos más próximo al punto que se muestran en el mapa.

Figura 7.5: Resultado de la búsqueda de los n barcos más próximos a un punto.

Por una parte, se pueden consultar las trayectorias de los barcos que pasaron por una región en un intervalo de tiempo. Para ello, el usuario tiene que presionar la tecla *ctrl* y arrastrar el ratón por el mapa seleccionando un área e indicar los instantes de tiempo de inicio y fin entre los cuales quiere obtener los puntos por los que pasaron los barcos. Una vez hecha la petición, se muestra en el mapa la región seleccionada por el usuario y la trayectoria de los barcos que pasaron por la región, como se puede ver en la Figura 7.8a. En esta imagen se puede ver claramente la trayectoria de un barco y para los demás se ve un punto, esto quiere decir que el barco se ha movido muy lentamente o nada en entre los instantes de tiempo que indicó el usuario. Además de visualizar en mapa las trayectorias, se muestra una tabla con los barcos resultado. De manera que el usuario puede acceder a la ficha de un barco o incluso ver la trayectoria del barco en concreto. Esto nos llevaría a una pantalla como la que se muestra en la Figura 7.3.

Por otra parte, se pueden consultar los barcos que pasaron por una región en un instante de

Figura 7.6: Ver la última posición de un barco concreto.

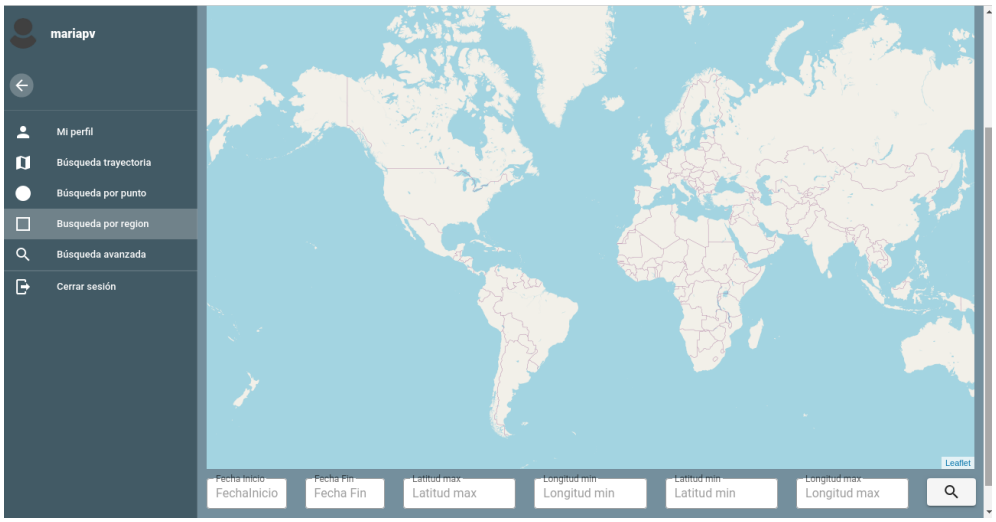
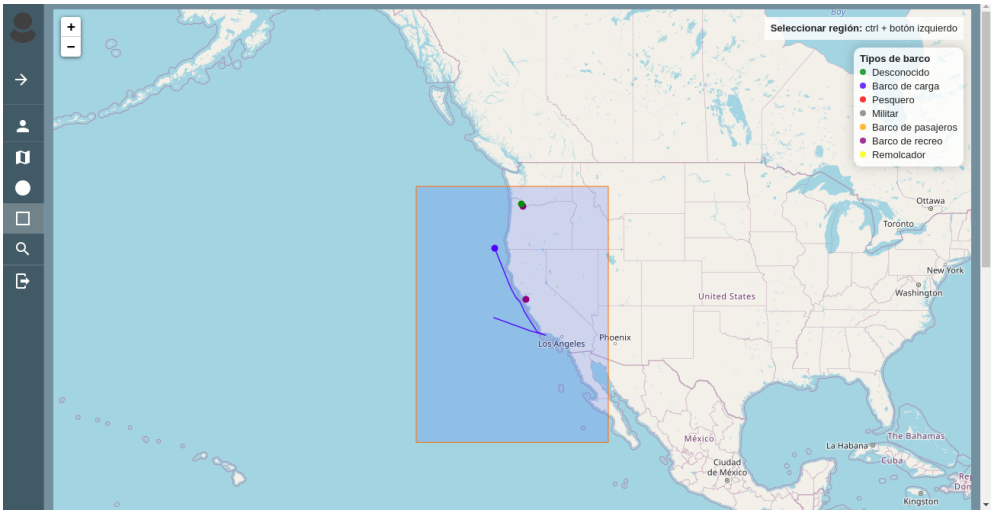
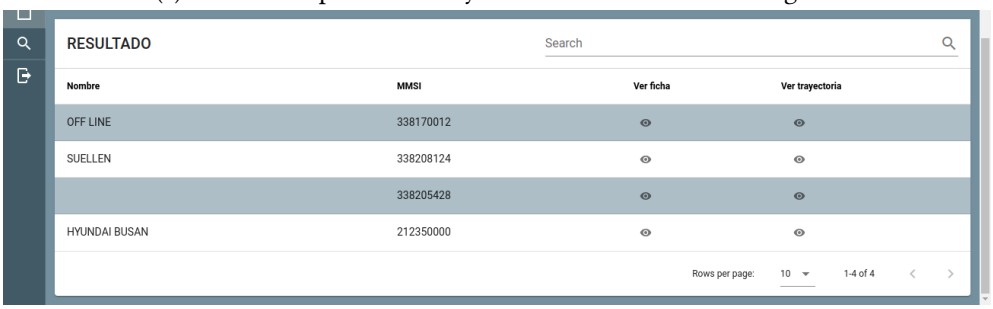


Figura 7.7: Consultar los barcos de una región.

tiempo concreto. Para hacer esta consulta, el usuario tiene que indicar un instante de tiempo, o introducir el mismo instante de tiempo de inicio y de fin, y seleccionar un área en el mapa del mismo modo que se explicó anteriormente. El resultado devuelto por la consulta se puede observar en la Figura 7.9a, en donde se visualiza en el mapa la última posición conocida de los barcos en el instante de tiempo y dentro de la región indicada por el usuario. Además, como en las consultas anteriores, se muestra una tabla con los barcos resultado de la consulta, ver Figura 7.9b, a través de la cual el usuario puede acceder tanto a la ficha de un barco concreto como a ver su última posición.



(a) Visor de mapas con la trayectoria de los barcos de la región.



(b) Tabla con los barcos que se muestran en el mapa.

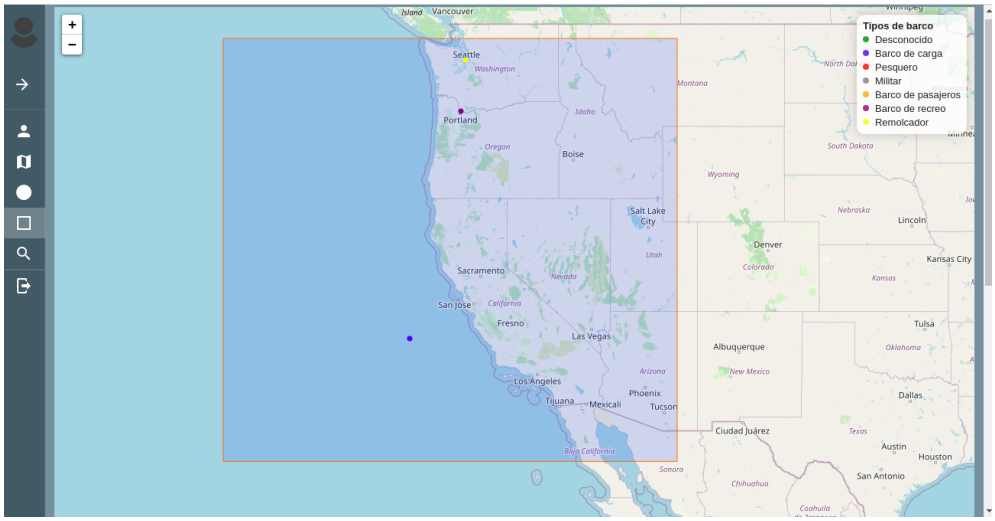
Figura 7.8: Resultado de la búsqueda de la trayectoria de los barcos de una región.

7.1.4 Búsqueda avanzada

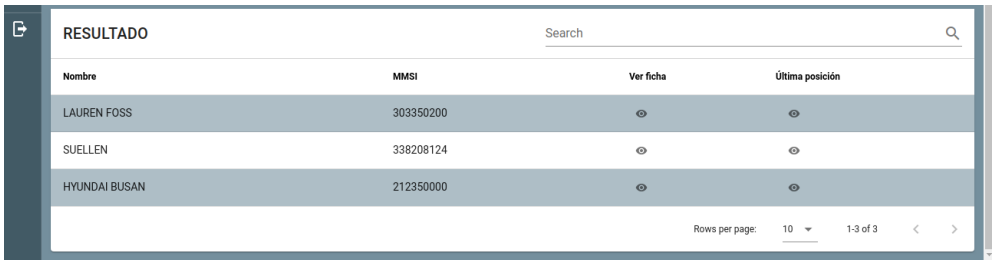
Por último, el usuario podrá realizar búsquedas a partir de las características de lo barcos, ver Figura 7.10. Una vez hecha la búsqueda, se muestra una tabla con los barcos resultado y el usuario tendrá la opción de mostrar en el mapa el resultado. De manera que, se visualiza en el visor de mapas la última posición de los barcos obtenidos en la búsqueda. Este resultado se mostrará en la pantalla principal, permitiendo al usuario realizar todas las operaciones que ahí están disponibles.

7.2 Acceso al perfil del usuario

Además de las consultas que el usuario puede hacer sobre los barcos, este puede acceder a su perfil, ver Figura 7.11, para ver la información almacenada, así como para hacer cambios en él, tanto en sus datos, como en su contraseña de acceso a la aplicación. Esta ficha incluye también una tabla que le permite ver al usuario los grupos a los que está asignado.



(a) Visor de mapas con la última posición de los barcos en una región determinada.



(b) Tabla con los barcos que se muestran en el mapa.

Figura 7.9: Resultado de la búsqueda de los barcos de un región en un instante de tiempo.

Figura 7.10: Buscador avanzado.

7.3 Gestión de usuarios, barcos y grupos

Además de las consultas explicadas en la Apartado 7.1 que pueden hacer todos los usuarios registrados, incluido el administrador, este se encarga a mayores de la gestión de los usuarios,

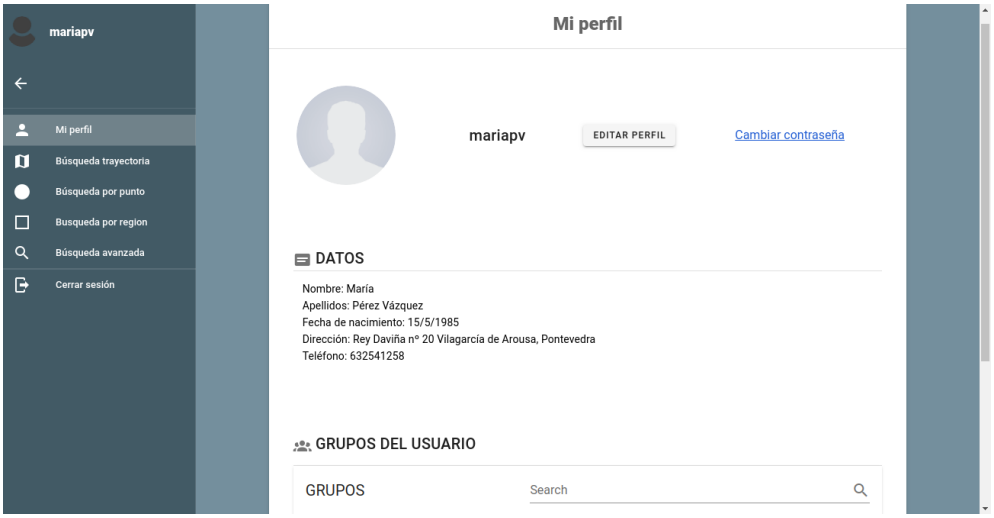


Figura 7.11: Perfil del usuario autenticado.

barcos y grupos.

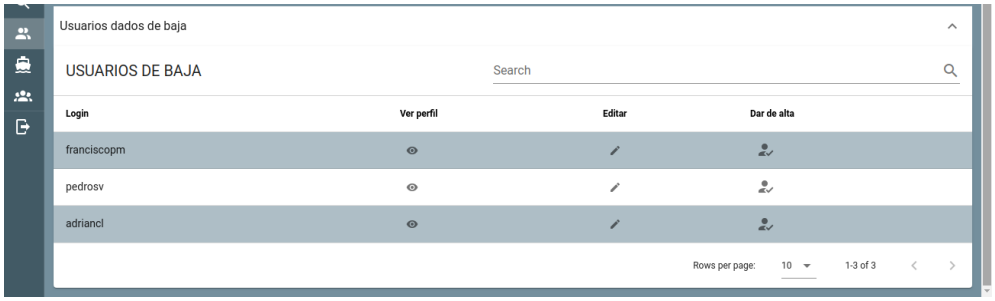
El usuario administrador tiene acceso a una lista con todos los usuarios que están registrados en la aplicación, ver Figura 7.12, en donde se distinguen dos listas. Una contiene los usuarios que están dados de alta en la aplicación Figura 7.12a, a través de la cual puede acceder al perfil de un usuario, editarlo y darlo de baja. Además, también dispone de un botón que le permite registrar más usuarios en la aplicación. La otra lista contiene los usuarios que están dados de baja y para cada uno de ellos puede acceder a su perfil, editarlo y también puede volver a darlo de alta en la aplicación, ver Figura 7.12b.

Existe una lista de barcos, ver la Figura 7.13, y en ella, el administrador, puede ejecutar operaciones como ver la ficha de un barco, editarla e incluso darlo de baja en la aplicación. Además tiene un botón para registrar un nuevo barco en la aplicación.

Para la gestión de los grupos, el administrador tiene acceso a una lista de grupos, ver Figura 7.14a, a través de la cual puede acceder a la información de un grupo concreto, en donde se muestra la lista de los usuarios y barcos que pertenecen a ese grupo. Desde esta pantalla el administrador puede quitar y añadir barcos y usuarios del grupo, como se puede ver en la Figura 7.14b.

Como se puede observar en las capturas de pantalla, todas las páginas siguen un estilo similar. De manera que sea cómodo para el usuario acceder a las funcionalidades disponibles. Además, tanto en la lista de usuarios, como de barcos y grupos, se le proporciona al administrador un buscador para facilitar el filtrado de estos.

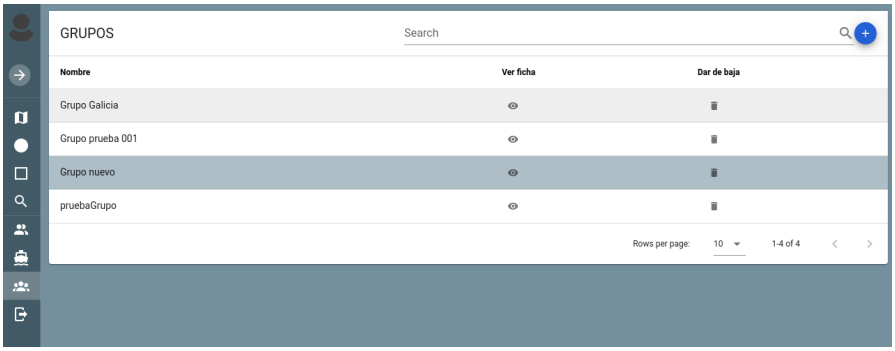
(a) Lista de los usuarios dados de alta en la aplicación.



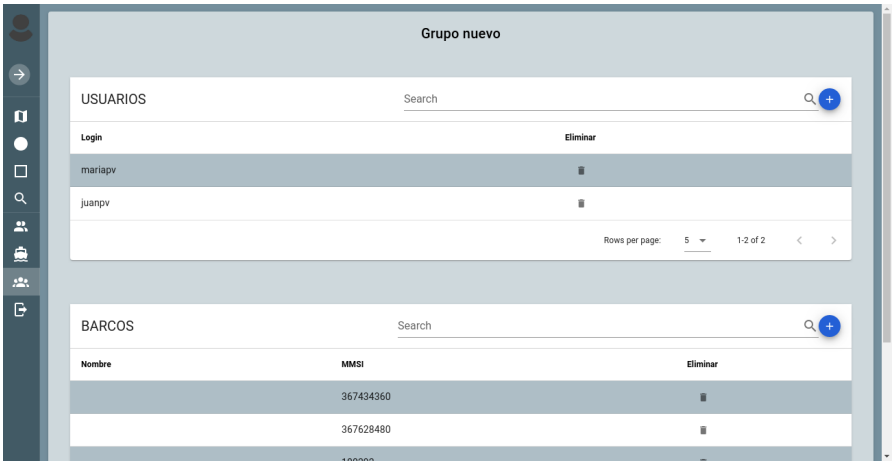
(b) Lista de los usuarios dados de baja en la aplicación.

Figura 7.12: Lista de usuarios.

Figura 7.13: Lista de barcos registrados en la aplicación.



(a) Lista de los grupos registrados en la aplicación.



(b) Ficha de un grupo concreto.

Figura 7.14: Grupos registrados.

Capítulo 8

Conclusiones y trabajo futuro

En esta sección se describe el grado en el que se han alcanzado los objetivos planificados inicialmente, las características más importantes y su utilidad en el mundo real y las lecciones aprendidas durante el desarrollo del proyecto. Así como también las ampliaciones futuras que pueden mejorar el proyecto.

8.1 Conclusiones finales

Una vez finalizado el proyecto se concluye lo siguiente:

- Los objetivos acordados en la planificación inicial del proyecto se han llevado a cabo, finalizando el desarrollo de la aplicación con todos y cada uno de ellos cumplidos exitosamente.
- Las características más importantes de esta aplicación son que el usuario pueda llevar a cabo la gestión de la flota permitiéndole realizar consultas de los barcos sobre datos históricos, como obtener su trayectoria, consultar los barcos que estuvieron más próximos a un punto de interés en un instante de tiempo, así como también obtener los barcos que pasaron por una región en un instante de tiempo y consultar la trayectoria de los barcos de una región en un intervalo de tiempo. Además, una de las características destacable es la integración de nuestra aplicación con el proyecto GraCT que almacena información geográfica de los barcos y permite hacer las consultas de manera muy eficiente obteniendo los resultados en breves instantes de tiempo. Por lo tanto, el desarrollo de este proyecto, proporciona al usuario una aplicación que le permite realizar consultas a cerca de los datos históricos de los barcos, a diferencia de las aplicaciones que hay actualmente en el mercado, y a su vez permite hacer uso de la estructura compacta GraCT y acercarla al mundo real.
- Durante el desarrollo del proyecto se han aprendido varias tecnologías que no se co-

nocían. Por ejemplo, el desarrollo del servicio con Express.js, así como la tecnología Leaflet, para la creación de mapas. También se aprendió a realizar la integración de un servicio con una estructura externa, utilizando nuevas tecnologías, C++ y N-API. Además, se han aplicado metodologías y conceptos, que se habían adquirido durante el grado: como es el uso de la metodología incremental, permitiendo la realización de cambios a medida que se avanzaba en el proyecto; el desarrollo de una aplicación utilizando la arquitectura en tres capas, permitiendo que se pudieran hacer modificaciones en una capa sin que afectara a las otras capas y realizar la planificación y seguimiento de un proyecto real. Se pudo comprobar que, debido a circunstancias no contempladas en la planificación, el proyecto se puede retrasar con respecto al tiempo estimado.

8.2 Posibles ampliaciones futuras

A continuación, se comentan algunas posibles ampliaciones del trabajo desarrollado que no eran parte del objetivo:

- Seguimiento real de los barcos. Actualmente, los datos de los barcos son datos históricos que se almacenan en GraCT y la última posición conocida que se almacena también en la base de datos conceptual de la aplicación que se actualiza una vez al día. Esta ampliación consistiría en que se actualizara la base de datos una vez que se registrara la última posición conocida del barco en la estructura GraCT.

- Conexión con otra base de datos. Conectar la aplicación con otra base de datos y realizar las consultas contra ella permitiendo comparar el tiempo que tardaría en resolver una consulta esta base de datos frente al tiempo que tardaría GraCT. Para ello, tendría que implementar una nueva clase con la misma interfaz para que se pueda conectar de manera transparente a la aplicación con algún fichero de configuración.

- Notificaciones de los reports. Enviar un aviso al dispositivo de los usuarios cuando alguien cree un comentario de un barco perteneciente a alguno de sus grupos.

- Internacionalización de la aplicación. Que pueda ser utilizada por un grupo más amplio de personas ya que estaría disponible en diferentes idiomas.

Apéndices

Apéndice A

Prototipos de pantalla

En este capítulo se muestra el diseño de los prototipos de pantalla de la aplicación, que debido a la falta de espacio no se pudieron añadir en la Apartado 4.3. Estas pantallas son prototipos y es posible que algunas de ellas no se correspondan exactamente con el diseño de la aplicación real. Todas las modificaciones llevadas a cabo fueron para proporcionarle al usuario una mejor interacción con el sistema.

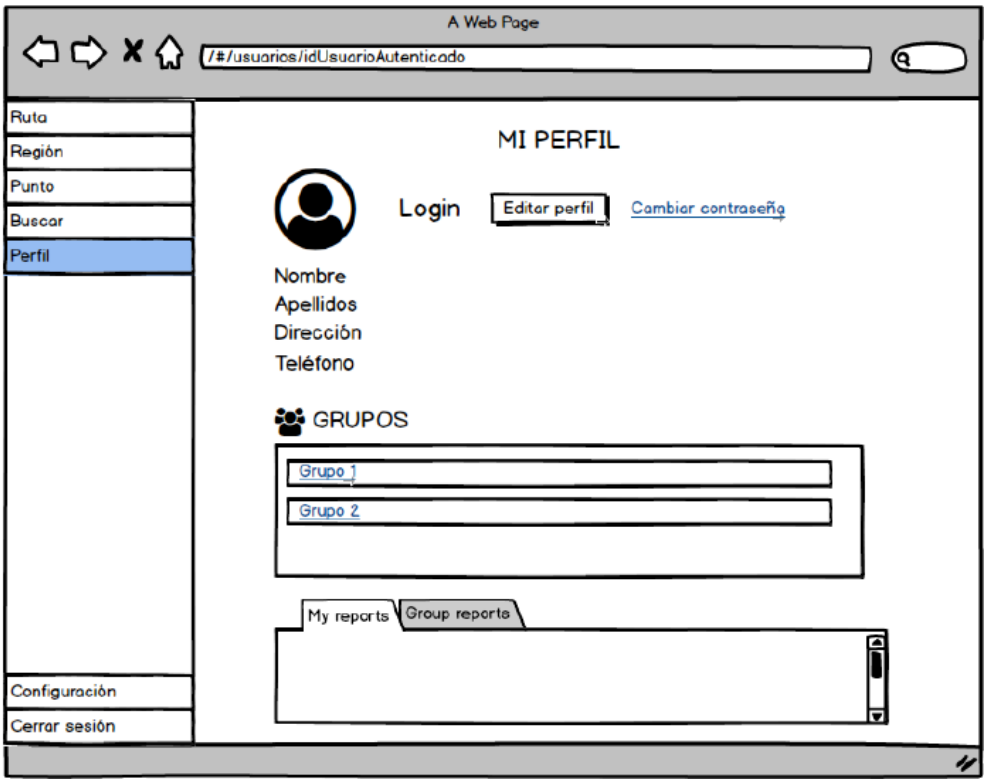


Figura A.1: Mockup del perfil del usuario autenticado.

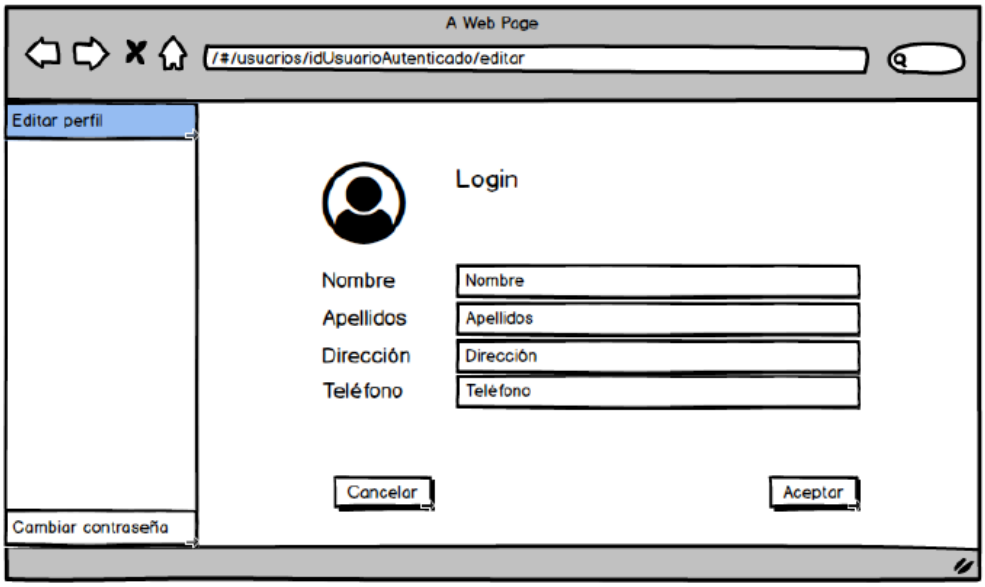


Figura A.2: Mockup para que el usuario autenticado edite su perfil.

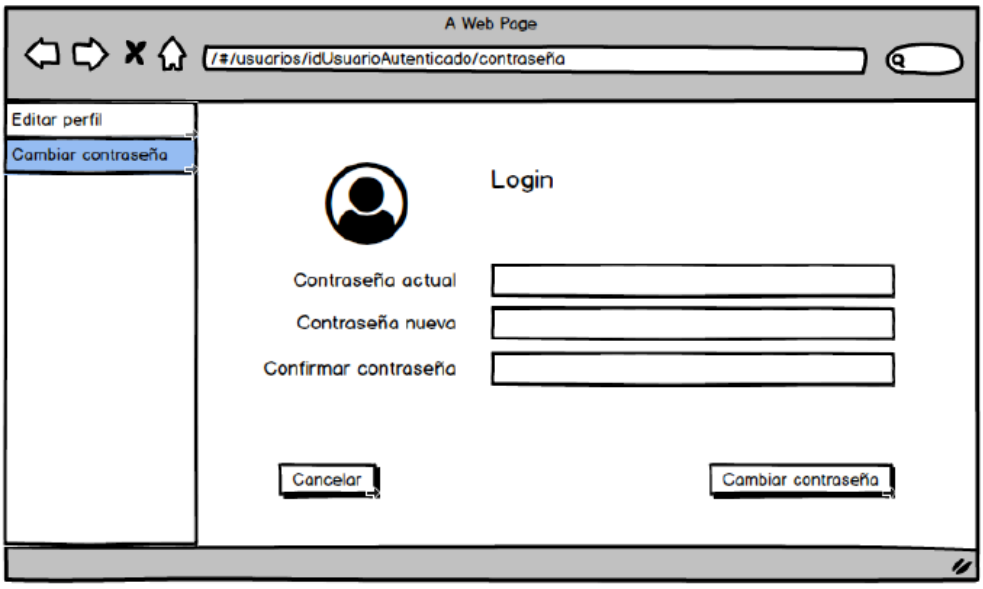


Figura A.3: Mockup para cambiar la contraseña.

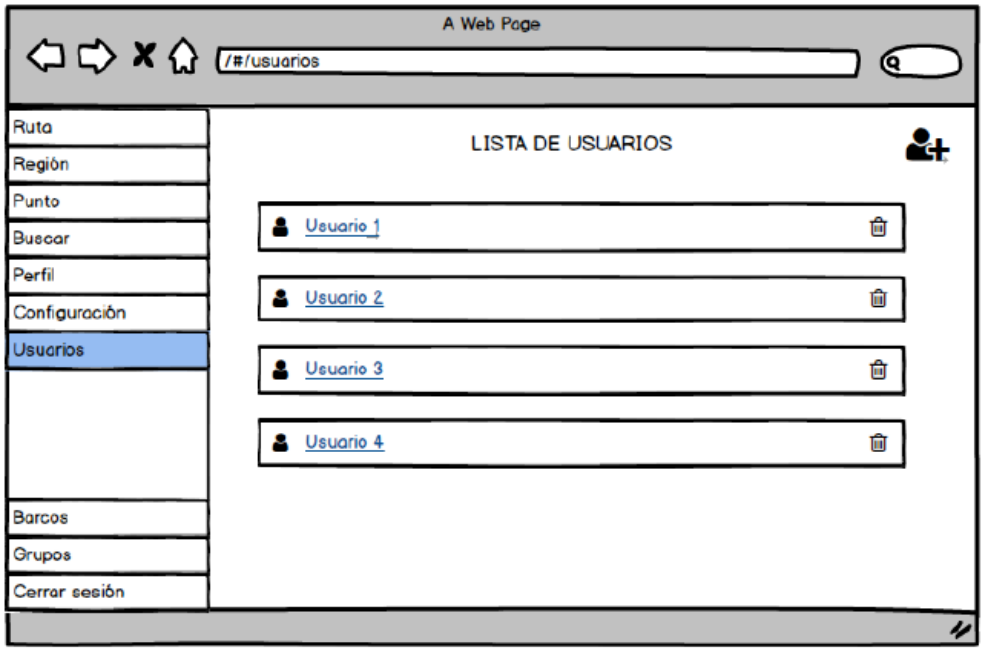


Figura A.4: Mockup de la lista de los usuarios registrados.

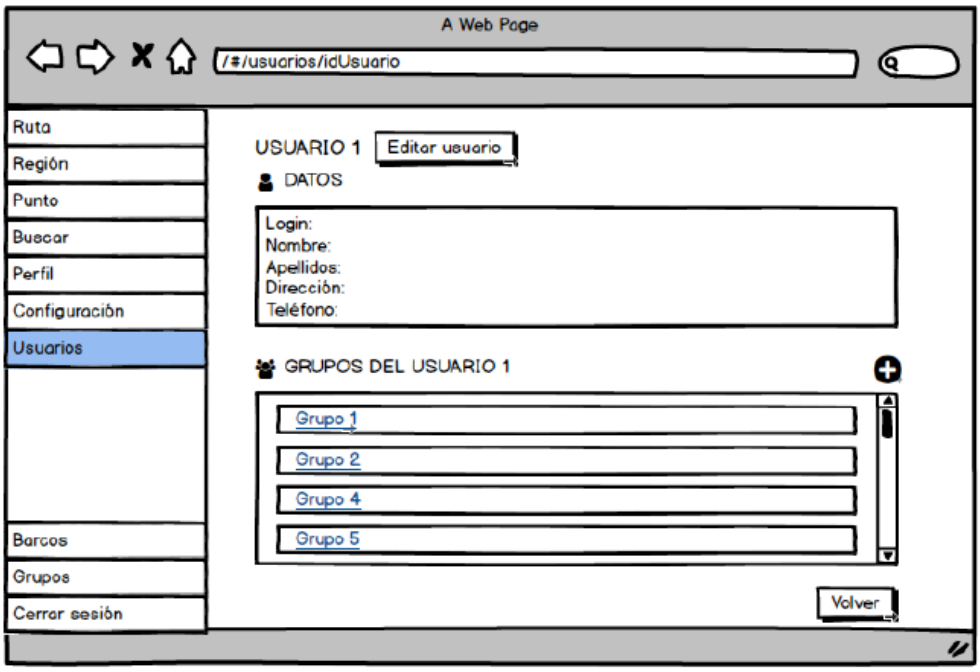


Figura A.5: Mockup del perfil de un usuario.

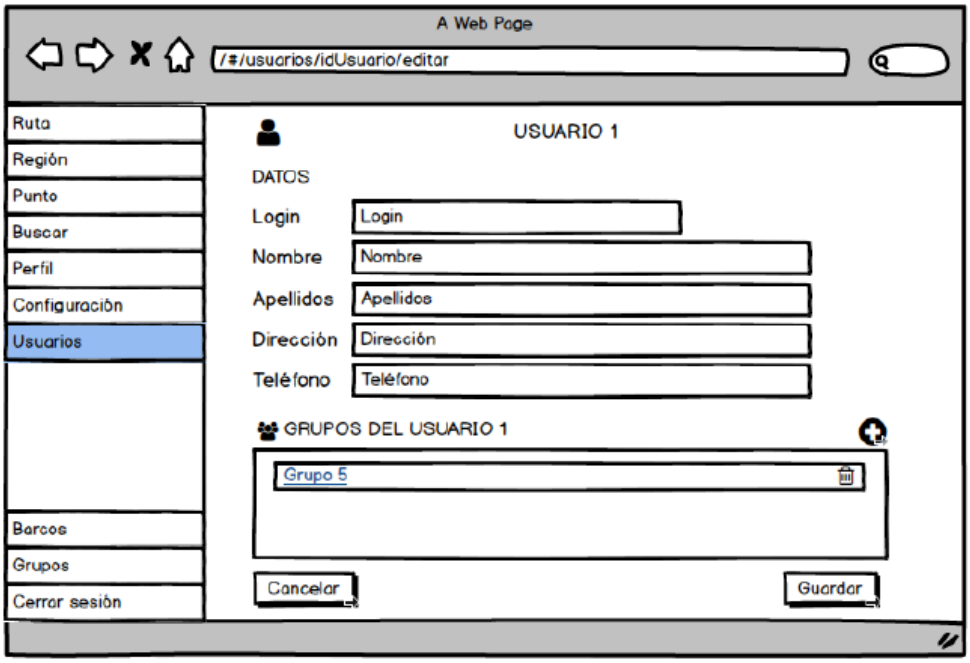


Figura A.6: Mockup para editar el perfil de un usuario.

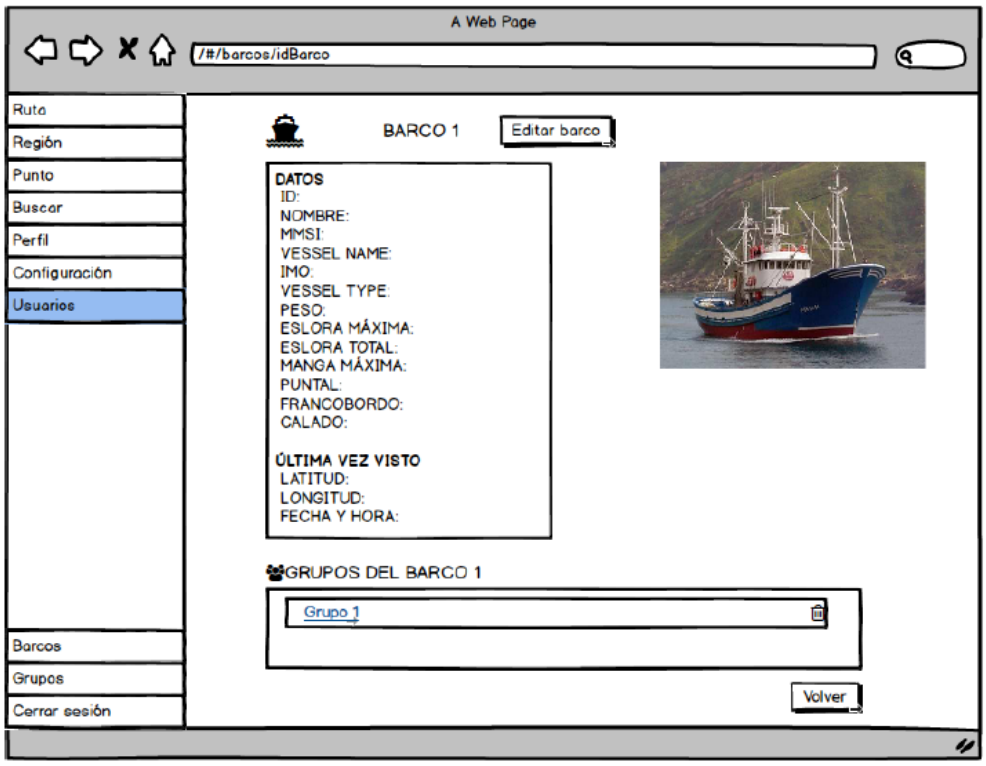


Figura A.7: Mockup de la ficha de un barco que ve el administrador.

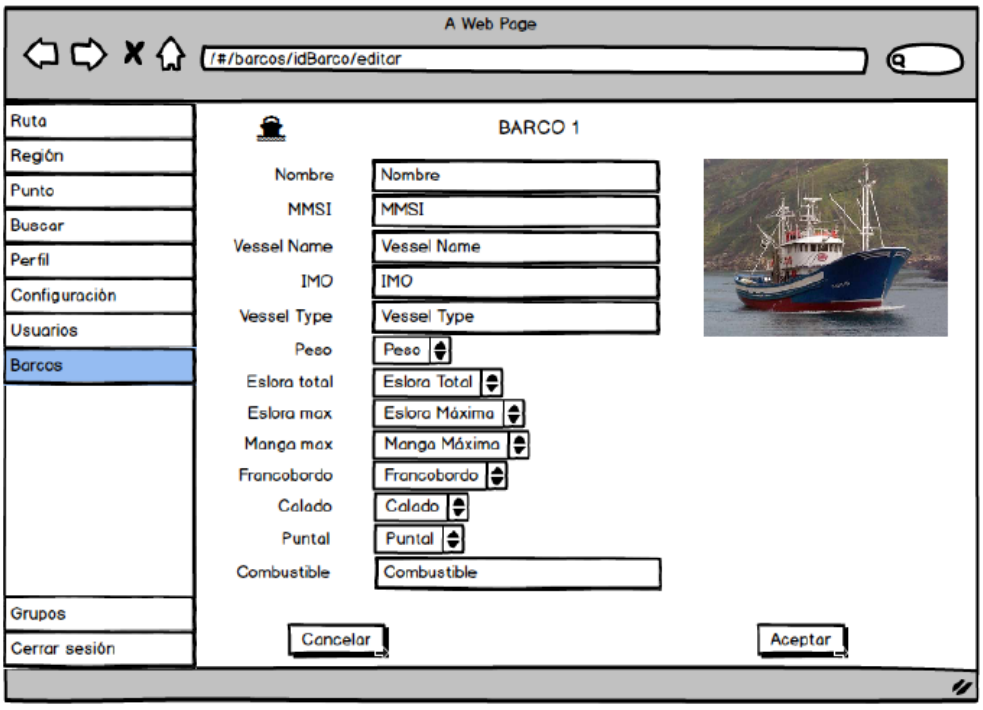


Figura A.8: Mockup para editar la ficha de un barco.

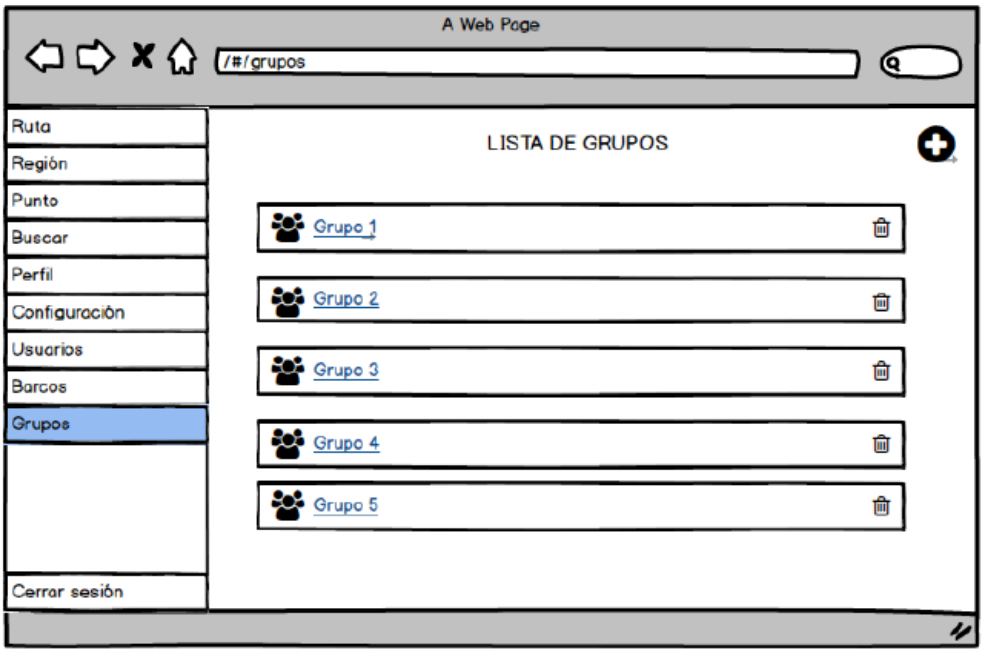


Figura A.9: Mockup de la lista de los grupos registrados.

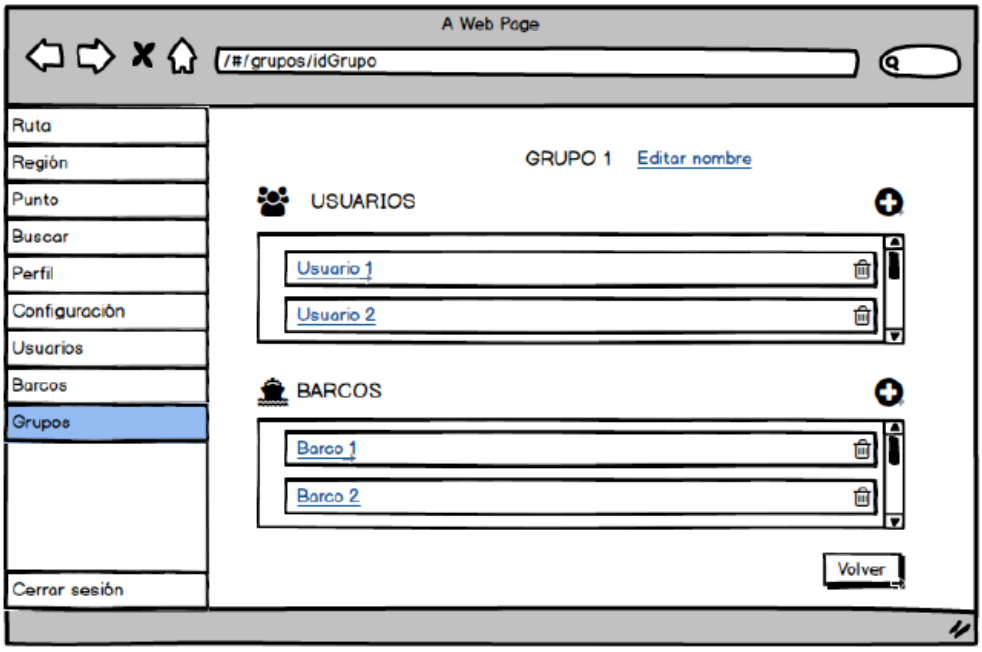


Figura A.10: Mockup de la ficha de un grupo.

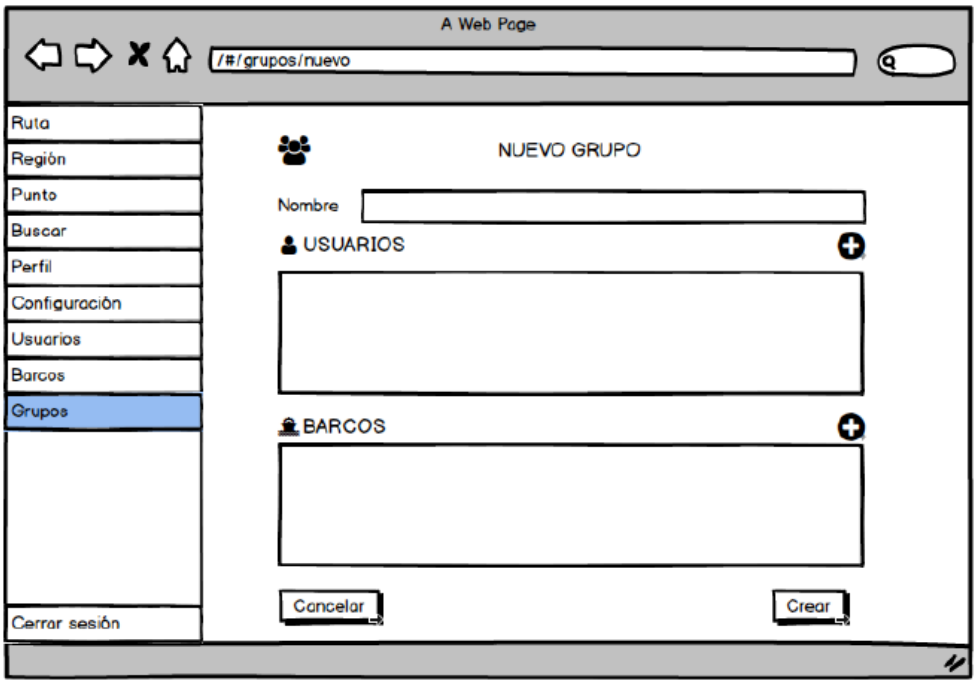


Figura A.11: Mockup para registrar un nuevo grupo en el sistema.
en el sistema

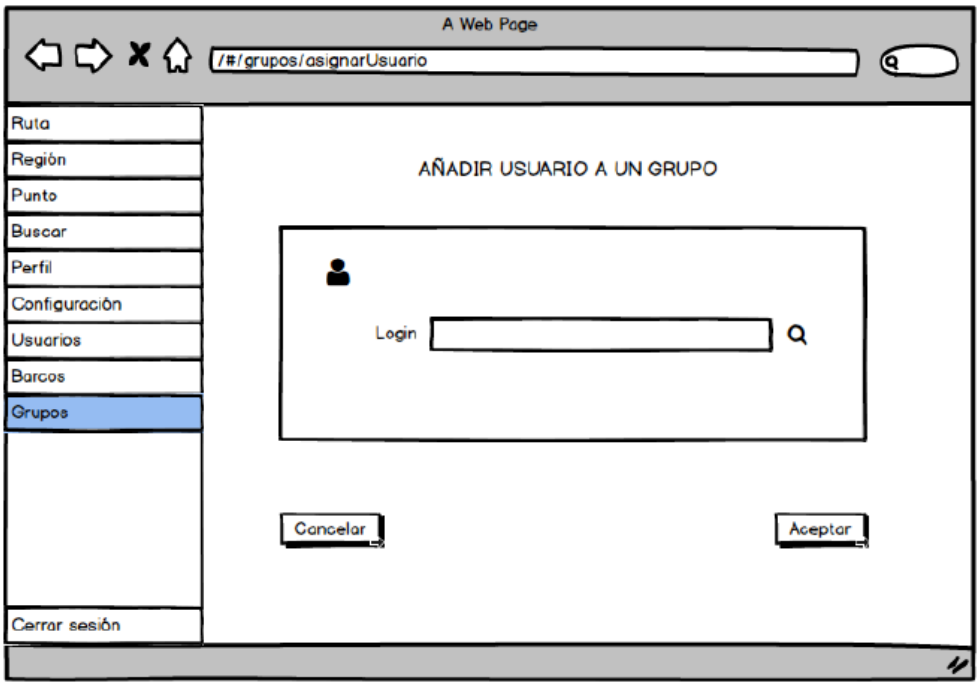


Figura A.12: Mockup para añadir un usuario a un grupo.

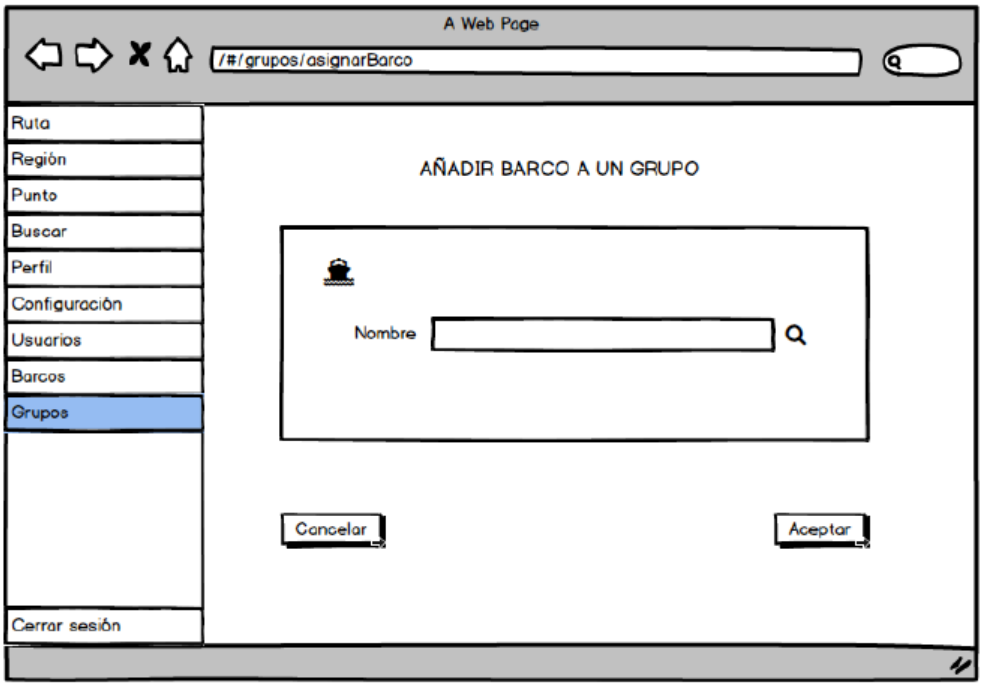


Figura A.13: Mockup para añadir un barco a un grupo.

Apéndice B

Manual de instalación

En este capítulo se detallan los pasos que hay que llevar a cabo para desplegar la aplicación en el dispositivo.

Previamente al despliegue de la aplicación es necesario tener instaladas en el sistema operativo las siguientes dependencias y librerías.

- Sistema de gestión de base de datos MongoDB.
- g++
- git
- node-js
- cmake
- npm
- node-gyp

  Estas dependencias se pueden instalar en Linux con los comandos *sudo apt install g++ git nodejs cmake npm* y *sudo npm install -g node-gyp*.
- A continuación se descarga la librería GraCT: *git clone https://gitlab.lbd.org.es/adriang-brandon/gractlib.git* y se instala *cd gractlib sudo ./install.sh*.
- Posteriormente, se inicia el proyecto **gract-module**: *npm init- y* y se instala la dependencia N-API con el comando *npm install -S node-addon-api*. Una vez iniciado el proyecto se compila con los siguientes comandos respectivamente: *node-gyp configure* y *node-gyp build*.
- Finalmente, se instalan las dependencias del **cliente** *npm install* y se ejecuta con *npm run serve*. Y se instalan las dependencias del **servicio** *npm install* y se ejecuta con *npm run start*.

Apéndice C

Glosario de acrónimos

AIS *Automatic Identification System.*

API *Application Programming Interfaces.*

BSON *Binary JavaScript Object Notation.*

CSV *Comma-Separated Values.*

CTRL *Control.*

DB *Data Base.*

GraCT *A Grammar based Compressed representation of Trajectories.*

HU *Historia de usuario.*

HTTP *Hypertext Transfer Protocol.*

IMO *International Maritime Organization.*

JSON *JavaScript Object Notation.*

MMSI *Maritime Mobile Service Identity.*

N-API *Node-Addon-Api.*

NAIS *Nationwide Automatic Identification System.*

NoSQL *Not only Structured Query Language.*

NPM *Node Package Manager.*

REST *Representational State Transfer.*

SGBD *Sistema de Gestión de Bases de Datos.*

UML *Unified Modeling Language.*

URL *Uniform Resource Locator.*

VHF *Very High Frequency.*

Apéndice D

Glosario de termos

Back-end Es la parte del software que se encarga de toda la lógica de negocio y del acceso a datos, conocido como servidor.

Endpoint Es la URL del servicio de un API que responde una petición.

Estructura compacta Es una estructura de datos que permite comprimir la información en el mínimo espacio posible sin perder eficiencia al consultar los datos.

Framework Es una estructura tecnológica que proporciona artefactos o módulos que facilitan el desarrollo software de una aplicación.

Front-end Es la parte del software que interactúa con el usuario, conocida como capa cliente.

Bibliografía

[1] N. R. Brisaboa, A. Gómez-Brandón, G. Navarro, and J. R. Paramá, "Gract: A grammar based compressed representation of trajectories," *CoRR*, vol. abs/1612.03308, 2016. [En línea]. Disponible en: http://arxiv.org/abs/1612.03308

[2] "Página web de leaflet," (consultado el 17/08/2020). [En línea]. Disponible en: https://leafletjs.com/

[3] "Página de vue.js," (consultado el 17/08/2020). [En línea]. Disponible en: https://vuejs.org/

[4] "Información de javascript," (consultado el 17/08/2020). [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript

[5] "Proyecto github de la librería axios," (consultado el 30/08/2020). [En línea]. Disponible en: https://github.com/axios/axios

[6] "Documentación de node.js," (consultado el 17/08/2020). [En línea]. Disponible en: https://nodejs.org/es/docs/

[7] "Página de express.js," (consultado el 17/08/2020). [En línea]. Disponible en: https://expressjs.com/es/

[8] "Página de mongodb," (consultado el 17/08/2020). [En línea]. Disponible en: https://www.mongodb.com/

[9] "Página de npm," (consultado el 30/08/2020). [En línea]. Disponible en: https://www.npmjs.com/

[10] "Página de n-api," (consultado el 30/08/2020). [En línea]. Disponible en: https://nodejs.org/api/n-api.html#n_api_n_api

[11] "Página de c++," (consultado el 17/08/2020). [En línea]. Disponible en: https://devdocs.io/cpp/

[12] A. G. Brandón, "Repositorio git de gract," (consultado el 17/08/2020). [En línea]. Disponible en: https://gitlab.lbd.org.es/adriangbrandon/gract

[13] "Información de scrum." [En línea]. Disponible en: https://obsbusiness.school/es/blog-investigacion/project-management/roles-eventos-y-artefactos-en-la-metodologia-scrum

[14] "Página de balsamiq," (consultado el 18/08/2020). [En línea]. Disponible en: https://balsamiq.com/wireframes/

[15] "Página de visual studio code," (consultado el 18/08/2020). [En línea]. Disponible en: https://code.visualstudio.com/

[16] "Página de gitlab," (consultado el 18/08/2020). [En línea]. Disponible en: http://about.gitlab.com/stages-devops-lifecycle/

[17] "Página de workbench," (consultado el 18/08/2020). [En línea]. Disponible en: https://www.mysql.com/products/workbench/

[18] "Página de postman," (consultado el 18/08/2020). [En línea]. Disponible en: https://www.postman.com/

[19] "Overleaf," (consultado el 18/08/2020). [En línea]. Disponible en: https://www.overleaf.com/project

[20] "Página de magicdraw," (consultado el 26/08/2020). [En línea]. Disponible en: https://www.nomagic.com/products/magicdraw

[21] "Página web de draw.io," (consultado el 02/09/2020). [En línea]. Disponible en: https://app.diagrams.net/

[22] "Página de dia," (consultado el 02/09/2020). [En línea]. Disponible en: https://wiki.gnome.org/Apps/Dia

[23] "Información de un raster," (consultado el 20/08/2020). [En línea]. Disponible en: https://en.wikipedia.org/wiki/Raster_data

[24] "Página donde se obtiene el data-set con la información geográfica de los barcos," (consultado el 01/09/2020). [En línea]. Disponible en: https://marinecadastre.gov/ais/

[25] "Guía para cargar código c++ en un proyecto de node.js de javascript," (consultado el 25/08/2020). [En línea]. Disponible en: https://medium.com/jspoint/a-simple-guide-to-load-c-c-code-into-node-js-javascript-applications-3fcccf54fd32